```
/* This routine creates a connection table
*/
void make_connection_table(int **bond_table, int *table_num,
                           rigid_unit *unit, rigid_unit *start)
{
  int i, *j, i1, save[MAX_BONDS];
  i1 = unit->head.atom_num + *table_num;
  for (j=unit->head.bond; *j != -1; j++) {
    add_connection(bond_table, i1, *j+*table_num);
    add_connection(bond_table, *j+*table_num, i1);
  }
  for (i=0; i<unit->n_bonds; i++) {
    i1 = unit->bond[i]->tail.atom_num + *table_num;
    for (j=unit->bond[i]->tail.bond; *j != -1; j++) {
      add_connection(bond_table, i1, *j+*table_num);
      add_connection(bond_table, *j+*table_num, i1);
    }
    save[i] = unit->bond[i]->tail.atom_num + *table_num;
  }
  *table_num += unit->n_atoms;
  for (i=0; i<unit->n_bonds; i++) {
    i1 = unit->bond[i]->next->head.atom_num;
    if (unit->bond[i]->next != start) i1 += *table_num;
    add_connection(bond_table, save[i], i1);
    add_connection(bond_table, i1, save[i]);
    if (unit->bond[i]->next != start)
       make_connection_table(bond_table,    table_num,
unit->bond[i]->next,start);
  }
}
/* This routine adds a connection to the connection table
*/
void add_connection(int **bond_table, int i1, int i2)
{
  int *i, *j;
  for (i=bond_table[i1]; *i != -1; i++) ;
  for (j=bond_table[i1]; j<i; j++) if (*j == i2) return;
  *i = i2;
}
```

```
/* This routine prints out the connection table
*/
void print_connection_table(int **bond_table, int n_atoms_total)
{
  int i, j;
  for (i=0; i<n_atoms_total; i++)  {
    printf("%5d     ",i);
    for (j=0; j<MAX_BONDS; j++) printf("%5d ", bond_table[i][j]);
    printf("\n");
  }
}
/* This routine determines the torsional terms
   p is set the head pointer and it returns the tail pointer
*/
void  get_torsions(torsion_list  **p,  int  **bond_table,  int
*table_num,
                   atom_list *atom, rigid_unit *unit, rigid_unit
*start)
{
  int i, save[MAX_BONDS];
  static torsion_list *q;
  static int i2, *j, *k;
  rigid_unit *new_unit;
  if (!*p) q = NULL;
  for (i=0; i<unit->n_bonds; i++)
    save[i] = unit->bond[i]->tail.atom_num + *table_num;
  *table_num += unit->n_atoms;
  for (i=0; i<unit->n_bonds; i++) {
    new_unit = unit->bond[i]->next;
    i2 = new_unit->head.atom_num;
    if (new_unit != start) i2 += *table_num;
    for (j=bond_table[save[i]]; *j != -1; j++)
      for (k=bond_table[i2]; *k != -1; k++)
        if (*j != i2 && save[i] != *k)
          if (!*p)
            *p = q = add_torsion(bond_table, atom, *j, save[i], i2,
*k);
          else
            if (q->next = add_torsion(bond_table, atom, *j,
```

```
                                                    save[i], i2, *k))
            q = q->next;
    if (new_unit != start)
       get_torsions(p, bond_table, table_num, atom, new_unit,
start);
  }
}
/* This routine adds a torsion to the torsion list
   Wildcards on i and l (simultaneously) are allowed for
*/
torsion_list *add_torsion(int **bond_table, atom_list *atom, int
i, int j,
                          int k, int l)
{
  torsion_list t, *v;
  char wild[]="*";
  int degen, itmp;
/* count degeneracy for "general" torsions--don't count the torsion
axis! */
/* "specific" torsions have a degeneracy of 1, "general" have a
degeneracy
    of degen */
  for (itmp=0; bond_table[j][itmp] != -1; itmp++) ;
  for (degen=0; bond_table[k][degen] != -1; degen++) ;
  itmp--;
  degen--;
  degen *= itmp;
  t.degen = 1;
/*  printf("%s %s %s %s %d\n",
                          atom[i].p->name,   atom[j].p->name,
atom[k].p->name,
                       atom[l].p->name, degen);  */
  t.next = NULL;
  t.num[0] = i;
  t.num[1] = j;
  t.num[2] = k;
  t.num[3] = l;
/* "specific" torsions */
  if   (!lookup_torsion_data(atom[i].p->type,   atom[j].p->type,
```

```
atom[k].p->type,
                        atom[l].p->type, &t.p)) {
/* "general" torsions */
    if    (!lookup_torsion_data(wild,    atom[j].p->type,
atom[k].p->type,
                            wild,& t.p)) {
          printf("Torsional data not found for %s %s %s %s\n",
                      atom[i].p->type,    atom[j].p->type,
atom[k].p->type,
                  atom[l].p->type);
          return(NULL);
        }
    t.degen = degen;
    }
/* only report nonzero torsional terms--this will screw up the 1/2
factor
   for AMBER! */
/*     if   (t.p->v0[0]==0   &&   t.p->v0[1]==0   &&   t.p->v0[2]==0)
return(NULL); */
  if ((v = (torsion_list *)
    malloc(sizeof(torsion_list))) == NULL) out_of_memory();
  *v = t;
  return(v);
}
/* This routine looks up the parameters for a torsional term in the
   torsion data base
*/
logical  lookup_torsion_data(string  type1,  string  type2,  string
type3,
                            string type4, torsion_data **p)
{
  torsion_data **l;
  for (l=torsion_data_list; *l; l++) {
    if   (strcmp((*l)->type1,   type1)==0   &&   strcmp((*l)->type2,
type2)==0 &&
            strcmp((*l)->type3,type3)==0    &&
strcmp((*l)->type4,type4)==0)
        goto done;
    if   (strcmp((*l)->type1,   type4)==0   &&   strcmp((*l)->type2,
```

```
type3)==0 &&
            strcmp((*l)->type3,type2)==0    &&
strcmp((*l)->type4,type1)==0)
       goto done;
  }
  return(FALSE);
done: ;
  *p = *l;
  return(TRUE);
}
/* This routine prints out the torsion terms
*/
void print_torsions(torsion_list *list, atom_list *atom)
{
  torsion_list *t;
  double theta;
  for (t=list; t; t=t->next)
  {
      theta    =    torsion(atom[t->num[0]].position,
atom[t->num[1]].position,
                           atom[t->num[2]].position,
atom[t->num[3]].position);
     printf("%4-s %4-s %4-s %4-s",atom[t->num[0]].p->name,
                           atom[t->num[1]].p->name,
                           atom[t->num[2]].p->name,
                           atom[t->num[3]].p->name);
/*         printf("%4-d  %4-d  %4-d  %4-d",t->num[0],  t->num[1],
t->num[2],
                             t->num[3]); */
     printf("%4d ",t->degen);
     printf("%9.3lf %7.3lf %7.3lf %7.3lf %7.3lf %7.3lf %7.3lf\n",
            180.0*theta/PI,
            t->p->v0[0], t->p->v0[1], t->p->v0[2],
            180.0*t->p->phi0[0]/PI, 180.0*t->p->phi0[1]/PI,
            180.0*t->p->phi0[2]/PI);
  }
}
/* This routine determines the torsional angle (in radians) defined
by the
```

```
   input positions--bonded in the order p1-p2-p3-p4
*/
double torsion(vector p1, vector p2, vector p3, vector p4)
{
  vector b1, b2, b3, n1, n2;
  double dot, len, theta;
/* define bond vectors */
  b3.x = p1.x - p2.x; b3.y = p1.y - p2.y; b3.z = p1.z - p2.z;
  b2.x = p3.x - p2.x; b2.y = p3.y - p2.y; b2.z = p3.z - p2.z;
  b1.x = p4.x - p3.x; b1.y = p4.y - p3.y; b1.z = p4.z - p3.z;
  b2 = vector_scale(b2, 1.0);
  dot = vector_dot(b1,b2);
/* project bonds onto torsion axis */
  n1.x = b1.x - dot*b2.x; n1.y = b1.y - dot*b2.y; n1.z = b1.z -
dot*b2.z;
  dot = vector_dot(b3,b2);
  n2.x = b3.x - dot*b2.x; n2.y = b3.y - dot*b2.y; n2.z = b3.z -
dot*b2.z;
  len = vector_length(n1)*vector_length(n2);
  theta = vector_dot(n1,n2)/len;
/* watch out for theta=0,PI, which kill acos */
  if (theta > 1.0-EPS)
    theta = 0.0;
  else if (theta < -1.0+EPS)
    theta = PI;
  else
    theta = acos(theta);
/* get proper sign on angle */
  n1 = vector_cross(n2,n1);
  if (vector_dot(n1, b2) < 0.0) theta = -theta;
  return(theta);
}
/* This function assigns the lennard jones parameters
*/
void assign_lj_parameters(rigid_unit *unit, rigid_unit *start)
{
  int i;
  for (i=0; i<unit->n_atoms; i++) {
    if (!lookup_lj_data(unit->atom[i].type, &unit->atom[i].ri,
```

```
          &unit->atom[i].ei)) {
        printf("Lennard-Jones parameters not found for atom %s\n",
                          unit->atom[i].type);
        exit(1);
      }
    }
    for (i=0; i<unit->n_bonds; i++)
      if (unit->bond[i]->next != start)
        assign_lj_parameters(unit->bond[i]->next, start);
}
/* This function looks up the lennard jones parameters for an atom
*/
logical lookup_lj_data(string type, double *ri, double *ei)
{
  lj_data **l;
  for (l=lj_data_list; *l; l++)
    if (strcmp((*l)->type, type)==0) {
      *ri = (*l)->ri;
      *ei = (*l)->ei;
      return(TRUE);
    }
  return(FALSE);
}
/* This routine determines the H-bonds that are in the molecule
*/
void get_hbonds(hbond_list **list, atom_list *atom, int n_atoms)
{
  int i,j;
  hbond_list t, *u, *v;
  *list = NULL;
  t.next = NULL;
  for (i=0; i<n_atoms; i++)
    for (j=i+1; j<n_atoms; j++)
      if   (lookup_hbond_data(atom[i].p->type,   atom[j].p->type,
&t.p)) {
        t.num[0] = i;
        t.num[1] = j;
        if ((v = (hbond_list *)
          malloc(sizeof(hbond_list))) == NULL) out_of_memory();
```

```
        *v = t;
        if (!*list)
          *list = u = v;
        else {
          u->next = v;
          u = u->next;
        }
      }
}
/* This function looks up the H-bond parameters for an atom pair
*/
logical lookup_hbond_data(string type1, string type2, hbond_data
**p)
{
  hbond_data **l;
  for (l=hbond_data_list; *l; l++) {
    if  (strcmp((*l)->type1,  type1)==0  &&  strcmp((*l)->type2,
type2)==0)
       goto done;
    if  (strcmp((*l)->type2,  type1)==0  &&  strcmp((*l)->type1,
type2)==0)
       goto done;
  }
  return(FALSE);
done: ;
  *p = *l;
  return(TRUE);
}
/* This function prints out the H-bonds
*/
void print_hbonds(hbond_list *l, atom_list *atom)
{
  for (; l; l=l->next) {
    printf("%s %s %lf %lf\n",
     atom[l->num[0]].p->name,  atom[l->num[1]].p->name,  l->p->a,
l->p->b);
  }
}
/* This function assigns the atom pointers
```

```
*/
void assign_atom_pointers(int *list_num, rigid_unit *unit,
rigid_unit *start,
                          atom_list *atom)
{
  int i;
  for (i=0; i<unit->n_atoms; i++) atom[i+*list_num].p =
&unit->atom[i];
  *list_num += unit->n_atoms;
  for (i=0; i<unit->n_bonds; i++)
    if (unit->bond[i]->next != start)
      assign_atom_pointers(list_num, unit->bond[i]->next, start,
atom);
}

**********************************************************************
            GEOMETRY CREATION ROUTINES - PEPTIDE3.C
**********************************************************************

/*                      The geometry creation routines
*/
#include "peptide.h"
logical grow_backwards=FALSE;
/* This function creates the Rosenbluth factor for an old
configuration
*/
void old_unit(int *list_num, int n0, int n1, int n2, double
*logrosen,
              rigid_unit *unit, rigid_unit *start, torsion_list *t,
              hbond_list *l, atom_list *atom, vector *twig[],
vector p0,
              vector b0)
{
  int i, j;
  vector p[MAX_BONDS], b[MAX_BONDS], p1, b1;
  double e;
  p1 = unit->atom[unit->head.atom_num].position;
  b1 = unit->head.axis;
  do_unit_sub(list_num, n0, n1, n2, logrosen, unit, t, l, atom,
```

```
twig,
                p1, b1, p0, b0, &e, p, b, FALSE);
  for (j=0; j<unit->n_bonds; j++)
    if (unit->bond[j]->next != start)
      old_unit(list_num, n0, n1, n2, logrosen, unit->bond[j]->next,
start,
                t, l, atom, twig, p[j], b[j]);
}
/* This function creates the geometry of a peptide
   and the Rosenbluth factor.  The growth is in one direction.
*/
void do_unit(int *list_num, int n0, int n1, int n2, double
*logrosen,
             rigid_unit *unit, rigid_unit *start, torsion_list *t,
             hbond_list *l, atom_list *atom, vector *twig[], vector
p0,
             vector b0, double *e)
{
  int i, j;
  vector p[MAX_BONDS], b[MAX_BONDS], p1, b1;
  unit->list_num = *list_num;
  p1 = unit->atom[unit->head.atom_num].position;
  b1 = unit->head.axis;
  do_unit_sub(list_num, n0, n1, n2, logrosen, unit, t, l, atom,
twig,
              p1, b1, p0, b0, e, p, b, TRUE);
/* loop over remaining units */
  for (j=0; j<unit->n_bonds; j++) {
/* store side-chain regrowth info */
    if (unit->bond[j]->next != start)
      do_unit(list_num, n0, n1, n2, logrosen,
              unit->bond[j]->next, start, t, l, atom, twig, p[j],
b[j], e);
  }
}
/* This function creates the geometry of a peptide
   and the Rosenbluth factor.  The growth is forward.
*/
void do_backbone_f(int i, int n_main, int n_atoms_total,
```

```
                    double *logrosen,
                    regrowth *main, regrowth *side,
                    torsion_list *t, hbond_list *l,
                    atom_list *atom, vector *twig[],
                    double *e, logical new)
{
  int list_num, n1, n2;
  vector p[MAX_BONDS], b[MAX_BONDS], p1, b1, p0, b0;
  if (i==0) i++;
  p0 = get_main_p0(atom, main, i);
  b0 = get_main_b0(atom, main, i);
  main += i;
  list_num = main->unit->list_num;
  n1 = n2 = n_atoms_total;
/* loop over backbone groups */
  for (; i<n_main; i++, main++) {
    p1 = main->unit->atom[main->unit->head.atom_num].position;
    b1 = main->unit->head.axis;
/* add on backbone unit */
    do_unit_sub(&list_num, 0, n1, n2, logrosen, main->unit, t, l,
atom, twig,
                p1, b1, p0, b0, e, p, b, new);
    if (!new && i < n_main-1) {
      p0 = get_main_p0(atom, main, 1);
      b0 = get_main_b0(atom, main, 1);
    } else if (new && i < n_main-1) {
      p0 = p[main->unit->n_bonds-1];
      b0 = b[main->unit->n_bonds-1];
    }
/* add on side chain */
    if (main->unit->n_bonds == 2) {
      if (new)
        do_unit(&list_num, 0, n1, n2, logrosen,
                        main->unit->bond[0]->next,
main->unit->bond[0]->next,
                t, l, atom, twig, p[0], b[0], e);
      else
        old_unit(&list_num, 0, n1, n2, logrosen,
                        main->unit->bond[0]->next,
```

```
main->unit->bond[0]->next,
                t, l, atom, twig, p[0], b[0]);
    }
  }
}
/* This function creates the geometry of a peptide
   and the Rosenbluth factor.  The growth is forward.
   Side chains are rigidly rotated.
*/
void do_backbone_f_rigid(int i, int n_main, int n_atoms_total,
                         double *logrosen,
                         regrowth *main, regrowth *side,
                         torsion_list *t, hbond_list *l,
                         atom_list *atom, atom_info *atom_tmp,
                         vector *twig[],
                         double *e, logical new)
{
  int list_num, n1, n2;
  vector p[MAX_BONDS], b[MAX_BONDS], p1, b1, p1a, b1a, p0, b0;
  logical false=FALSE;

  int n_atoms, j;
  atom_info *q;
  double len;
  vector b2[MAX_BONDS], v, v2;
  if (i==0) i++;
  p0 = get_main_p0(atom, main, i);
  b0 = get_main_b0(atom, main, i);
  main += i;
  list_num = main->unit->list_num;
  n1 = n2 = n_atoms_total;
/* get first head vector */
   p1   =   atom[main->unit->list_num   +
main->unit->head.atom_num].position;
  b1 = atom[main[-1].unit->list_num +

main[-1].unit->bond[main[-1].unit->n_bonds-1]->tail.atom_num]
          .position;
  b1.x = p1.x - b1.x;
```

```
  b1.y = p1.y - b1.y;
  b1.z = p1.z - b1.z;
  for (; i<n_main; i++, main++) {
/* change unit */
    n_atoms = main->unit->n_atoms;
    q = main->unit->atom;
    if (i < n_main-1)
      main->unit->n_atoms = main[1].unit->list_num -
main->unit->list_num;
    main->unit->atom = atom_tmp;
    for (j=0; j<main->unit->n_atoms; j++)
      main->unit->atom[j].position = atom[list_num+j].position;
    for (j=0; j<main->unit->n_bonds; j++) {
      b2[j] = main->unit->bond[j]->tail.axis;
      v = atom[main->unit->bond[j]->next->list_num +
               main->unit->bond[j]->next->head.atom_num].position;
      v2 = atom[main->unit->list_num +
                main->unit->bond[j]->tail.atom_num].position;
      v.x -= v2.x;
      v.y -= v2.y;
      v.z -= v2.z;
      main->unit->bond[j]->tail.axis = vector_scale(v,1.0);
    }
/* get next head vector */
    if (i < n_main-1) {
      p1a = atom[main[1].unit->list_num +
                 main[1].unit->head.atom_num].position;
      b1a = atom[main->unit->list_num +

main->unit->bond[main->unit->n_bonds-1]->tail.atom_num]
               .position;
      b1a.x = p1a.x - b1a.x;
      b1a.y = p1a.y - b1a.y;
      b1a.z = p1a.z - b1a.z;
    }
/* add on unit */
    do_unit_sub(&list_num, 0, n1, n2, logrosen, main->unit, t, 1,
atom, twig,
                p1, b1, p0, b0, e, p, b, new);
```

```
/* change unit back */
    main->unit->n_atoms = n_atoms;
    main->unit->atom = q;
    for (j=0; j<main->unit->n_bonds; j++)
      main->unit->bond[j]->tail.axis = b2[j];
/* change head vector */
    if (!new && i < n_main-1) {
      p0 = get_main_p0(atom, main, 1);
      b0 = get_main_b0(atom, main, 1);
    } else if (new && i < n_main-1) {
      p0 = p[main->unit->n_bonds-1];
      b0 = b[main->unit->n_bonds-1];
    }
    p1 = p1a;
    b1 = b1a;
  }
}
/* This function creates the geometry of a peptide
   and the Rosenbluth factor.  The growth is backward.
*/
void do_backbone_b(int i, int n_main, int n_atoms_total,
                   double *logrosen,
                   regrowth *main, regrowth *side,
                   torsion_list *t, hbond_list *l,
                   atom_list *atom, vector *twig[],
                   double *e, logical new)
{
  int list_num, n0, n1, n2, n_bonds;
  vector p[MAX_BONDS], b[MAX_BONDS], b0, p0, tmp, p1, b1;
  if (i == n_main-1) i--;
  main += i;
  n2 = n_atoms_total;
  b0 = get_main_b0(atom, main, 1);
  for (; i>=0; i--, main--) {
    n1 = main[1].unit->list_num;
    n0 = list_num = main->unit->list_num;
/* get bond vectors */
                                    p       0                          =
atom[main[1].unit->head.atom_num+main[1].unit->list_num].position;
```

```
    b0.x = -b0.x; b0.y = -b0.y; b0.z = -b0.z;
    n_bonds = main->unit->n_bonds;
    p1 = main->unit->atom[main->unit->bond[n_bonds-1]->
                  tail.atom_num].position;
    b1 = main->unit->bond[n_bonds-1]->tail.axis;
    b1.x = -b1.x;
    b1.y = -b1.y;
    b1.z = -b1.z;
    b1 = vector_scale(b1, vector_length(main[1].unit->head.axis));
    tmp = main->unit->bond[n_bonds-1]->tail.axis;
    main->unit->bond[n_bonds-1]->tail.axis = main->unit->head.axis;
/* add on unit */
    grow_backwards = TRUE;
    do_unit_sub(&list_num, n0, n1, n2, logrosen, main->unit, t, l,
atom, twig,
                p1, b1, p0, b0, e, p, b, new);
    grow_backwards = FALSE;
    main->unit->bond[n_bonds-1]->tail.axis = tmp;
/* change head vector */
    if (!new && i > 0)
      b0 = get_main_b0(atom, main-1, 1);
    else if (new && i > 0)
      b0 = vector_scale(b[n_bonds-1], 1.0);
/* add on side chain */
    if (main->unit->n_bonds == 2) {
      if (new)
        do_unit(&list_num, n0, n1, n2, logrosen,
                        main->unit->bond[0]->next,
main->unit->bond[0]->next,
                t, l, atom, twig, p[0], b[0], e);
      else
        old_unit(&list_num, n0, n1, n2, logrosen,
                         main->unit->bond[0]->next,
main->unit->bond[0]->next,
                 t, l, atom, twig, p[0], b[0]);
    }
  }
}
/* This function creates the geometry of a peptide
```

```
   and the Rosenbluth factor.  The growth is backward.
   Side chains are rigidly rotated.
*/
void do_backbone_b_rigid(int i, int n_main, int n_atoms_total,
                         double *logrosen,
                         regrowth *main, regrowth *side,
                         torsion_list *t, hbond_list *l,
                         atom_list  *atom,  atom_info  *atom_tmp,
vector *twig[],
                         double *e, logical new)
{
  int list_num, n0, n1, n2, n_bonds, n_atoms, j;
  vector p[MAX_BONDS], b[MAX_BONDS], b0, p0, tmp, p1, b1, p1a, b1a,
         b2[MAX_BONDS], v, v2;
  logical false=FALSE;
  atom_info *q;
  if (i == n_main-1) i--;
  main += i;
  n2 = n_atoms_total;
/* get first head unit */
  p1 = atom[main->unit->bond[main->unit->n_bonds-1]->tail.atom_num
+
            main->unit->list_num].position;
   b1    =    atom[main[1].unit->list_num    +
main[1].unit->head.atom_num].position;
  b1.x = p1.x - b1.x;
  b1.y = p1.y - b1.y;
  b1.z = p1.z - b1.z;
  b0 = get_main_b0(atom, main, 1);
  for (; i>=0; i--, main--) {
/* get current info */
    list_num = main->unit->list_num;
    n_bonds = main->unit->n_bonds;
                                  p       0                    =
atom[main[1].unit->head.atom_num+main[1].unit->list_num].position;
    b0.x = -b0.x; b0.y = -b0.y; b0.z = -b0.z;
    n1 = main[1].unit->list_num;
    n0 = list_num = main->unit->list_num;
    n_atoms = main->unit->n_atoms;
```

```
    q = main->unit->atom;
/* change current unit */
    main->unit->n_atoms = n1 - n0;
    main->unit->atom = atom_tmp;
    for (j=0; j<main->unit->n_atoms; j++)
      main->unit->atom[j].position = atom[list_num+j].position;
/* compute bond axes */
    for (j=0; j<n_bonds; j++) {
      b2[j] = main->unit->bond[j]->tail.axis;
      v = atom[main->unit->bond[j]->next->list_num +
               main->unit->bond[j]->next->head.atom_num].position;
      v2 = atom[list_num +
                main->unit->bond[j]->tail.atom_num].position;
      v.x -= v2.x;
      v.y -= v2.y;
      v.z -= v2.z;
      main->unit->bond[j]->tail.axis = vector_scale(v,1.0);
    }
    main->unit->bond[n_bonds-1]->tail.axis =
        vector_scale(get_main_b0(atom, main-i, i),
                     vector_length(main->unit->head.axis));
/* compute new head vector */
    if (i > 0) {
                                        p     1     a                =
atom[main[-1].unit->bond[main[-1].unit->n_bonds-1]->tail.atom_num+
                main[-1].unit->list_num].position;
      b1a=atom[list_num + main->unit->head.atom_num].position;
      b1a.x = p1a.x - b1a.x;
      b1a.y = p1a.y - b1a.y;
      b1a.z = p1a.z - b1a.z;
    }
/* add on unit */
    grow_backwards = TRUE;
    do_unit_sub(&list_num, n0, n1, n2, logrosen, main->unit, t, 1,
atom, twig,
                p1, b1, p0, b0, e, p, b, new);
    grow_backwards = FALSE;
/* restore backbone unit */
    main->unit->n_atoms = n_atoms;
```

```
    main->unit->atom = q;
    for (j=0; j<n_bonds; j++) {
      main->unit->bond[j]->tail.axis = b2[j];
/* change head vector */
    if (!new && i > 0)
      b0 = get_main_b0(atom, main-1, 1);
    else if (new && i > 0)
      b0 = vector_scale(b[n_bonds-1], 1.0);
    p1 = p1a;
    b1 = b1a;
    }
  }
}
/* This routine creates the random positions.
   For new units, it picks and copies the winner.
*/
void do_unit_sub(int *list_num, int n0, int n1, int n2, double
*logrosen,
                 rigid_unit *unit, torsion_list *t, hbond_list *l,
                 atom_list *atom, vector *twig[], vector p1, vector
b1,
                 vector p0, vector b0, double *e, vector
p[MAX_BONDS],
                 vector b[MAX_BONDS], logical new)
{
  int i,j,i0;
  vector bond[KMAX][MAX_BONDS], point[KMAX][MAX_BONDS];
  double ftmp, cos_theta2, sin_theta2;
  double de[KMAX], sum, max;
  i0 = 0;
  if (!new) {
/* copy old configuration to first "guess" */
    i0 = 1;
    for (j=0; j<unit->n_atoms; j++)
      twig[0][j] = atom[*list_num + j].position;
  }
/* create gueses for new unit position */
  for (i=i0; i<KMAX; i++) {
    do {
```

```
      cos_theta2 = 1-2*ran2(1.0);
      sin_theta2 = 1-2*ran2(1.0);
      ftmp = cos_theta2*cos_theta2 + sin_theta2*sin_theta2;
    } while (ftmp > 1.0);
    ftmp = sqrt(ftmp);
    cos_theta2 /= ftmp;
    sin_theta2 /= ftmp;
    add_rigid_unit(unit, twig[i], p1, b1,
                   p0, b0, point[i], bond[i],
                   cos_theta2, sin_theta2);
  }
/* calculate probabilties -- be careful about zero of energy &
overflows */
  max = -1E99;
  for (j=0; j<KMAX; j++) {
    de[j] = -BETA * delta_energy(t, 1, atom, twig[j], *list_num,
n0, n1, n2,
                                 unit->n_atoms);
    if (de[j] > max) max = de[j];
  }
  sum = 0.0;
  for (j=0; j<KMAX; j++) {
    de[j] = exp(de[j] - max);
    sum += de[j];
  }
  *logrosen += log(sum) + max - log(KMAX);
  if (!new) {
/* determine points */
    for (j=0; j<unit->n_bonds; j++) {
          p[j] = atom[*list_num +
unit->bond[j]->tail.atom_num].position;
      b[j] = atom[unit->bond[j]->next->list_num +
                  unit->bond[j]->next->head.atom_num].position;
      b[j].x -= p[j].x;
      b[j].y -= p[j].y;
      b[j].z -= p[j].z;
      b[j] = vector_scale(b[j], 1.0);
    }
    *list_num += unit->n_atoms;
```

```
  } else {
/* pick winner */
    de[0] /= sum;
    for (j=1; j<KMAX; j++) de[j] = de[j-1] + de[j]/sum;
    ftmp = ran2(1.0);
    for (i=0; i<KMAX; i++) if (ftmp <= de[i]) break;
    ftmp = de[i];
    if (i > 0) ftmp -= de[i-1];
    ftmp *= sum;
    *e -= (log(ftmp)+max)/BETA;
/* copy winner to atom array */
    for (j=0; j<unit->n_atoms; j++, (*list_num)++)
       atom[*list_num].position = twig[i][j];
    for (j=0; j<unit->n_bonds; j++) {
      p[j] = point[i][j];
      b[j] = bond[i][j];
    }
  }
}
/* This routine adds a rigid unit to the peptide structure
*/
void add_rigid_unit(rigid_unit *unit, vector *pos,
                    vector p1, vector b1, vector p0,
                    vector b0, vector point[MAX_BONDS],
                    vector bond[MAX_BONDS],
                    double cos_theta2, double sin_theta2)
{
  int i;
  double bond_len, cos_theta, sin_theta;
  vector n, r0;
  bond_len = vector_length(b1);
  r0.x = p0.x + b0.x*bond_len;
  r0.y = p0.y + b0.y*bond_len;
  r0.z = p0.z + b0.z*bond_len;
  b1.x /= bond_len;
  b1.y /= bond_len;
  b1.z /= bond_len;
  n = vector_cross(b1,b0);
  cos_theta = vector_dot(b0,b1);
```

```
    sin_theta = vector_length(n);
    if (sin_theta < EPS) {
      n.x = 1.0;
    } else {
      n.x /= sin_theta;
      n.y /= sin_theta;
      n.z /= sin_theta;
    }
    for (i=0; i<unit->n_atoms; i++)
      pos[i] = align(unit->atom[i].position, r0, p1,
                     n, cos_theta, sin_theta,
                     b0, cos_theta2, sin_theta2);
    for (i=0; i<unit->n_bonds; i++)
      point[i] = pos[unit->bond[i]->tail.atom_num];
    r0.x = 0.0; r0.y = 0.0; r0.z = 0.0; p1=r0;
    for (i=0; i<unit->n_bonds; i++)
      bond[i] = align(unit->bond[i]->tail.axis, r0, p1,
                      n, cos_theta, sin_theta,
                      b0, cos_theta2, sin_theta2);
}
/* This routine aligns the position
*/
vector align(vector p, vector r0, vector r1, vector n, double
cos_theta,
             double sin_theta, vector n2, double cos_theta2,
             double sin_theta2)
{
  vector ret;
  ret.x = p.x - r1.x;
  ret.y = p.y - r1.y;
  ret.z = p.z - r1.z;
  ret = vector_rotate(ret, n, cos_theta, sin_theta);
  ret = vector_rotate(ret, n2, cos_theta2, sin_theta2);
  ret.x += r0.x;
  ret.y += r0.y;
  ret.z += r0.z;
  return(ret);
}
```

```
*************************************************************************
                ENERGY DETERMINATION - PEPTIDE4.C
*************************************************************************

/*                          The energy routines
*/
#include "peptide.h"
#define N0 8
#define N1 11
#define N2 81
#define N3 84
#define N2 63
#define N3 66
#define SCALE 100
/* This energy routine tries to force a S-S ring-closure for
CAAAAAAC
*/
double zenergy(torsion_list *t, hbond_list *l, atom_list *atom,
               int n_atoms_total)
{
  double r1, r2;
  vector x, y, v;
  x = atom[N1].position;
  x.x -= atom[N0].position.x;
  x.y -= atom[N0].position.y;
  x.z -= atom[N0].position.z;
  x = vector_scale(x, 2.038);
  x.x += atom[N0].position.x;
  x.y += atom[N0].position.y;
  x.z += atom[N0].position.z;
  y = atom[N3].position;
  y.x -= atom[N2].position.x;
  y.y -= atom[N2].position.y;
  y.z -= atom[N2].position.z;
  y = vector_scale(y, 2.038);
  y.x += atom[N2].position.x;
  y.y += atom[N2].position.y;
  y.z += atom[N2].position.z;
  v = x;
```

```
    v.x -= atom[N2].position.x;
    v.y -= atom[N2].position.y;
    v.z -= atom[N2].position.z;
    r1 = vector_length2(v);
    v = y;
    v.x -= atom[N0].position.x;
    v.y -= atom[N0].position.y;
    v.z -= atom[N0].position.z;
    r2 = vector_length2(v);
    return(SCALE*(r1+r2)/BETA);
}
/* This energy routine tries to force a S-S ring-closure for
CAAAAAAC
*/
double zdelta_energy(torsion_list *t, hbond_list *l, atom_list
*atom,
                     vector *twig, int n_atoms, int n0, int n1, int
n2,
                     int n_twig)
{
  double r1, r2;
  vector x, y, v;
  r1 = r2 = 0.0;
  if (INTERVAL(N0, n_atoms, n_atoms+n_twig) &&
      INTERVAL(N2, n1, n2)) {
    x = twig[N1-n_atoms];
    x.x -= twig[N0-n_atoms].x;
    x.y -= twig[N0-n_atoms].y;
    x.z -= twig[N0-n_atoms].z;
    x = vector_scale(x, 2.038);
    x.x += twig[N0-n_atoms].x;
    x.y += twig[N0-n_atoms].y;
    x.z += twig[N0-n_atoms].z;
    y = atom[N3].position;
    y.x -= atom[N2].position.x;
    y.y -= atom[N2].position.y;
    y.z -= atom[N2].position.z;
    y = vector_scale(y, 2.038);
    y.x += atom[N2].position.x;
```

```
    y.y += atom[N2].position.y;
    y.z += atom[N2].position.z;
    v = x;
    v.x -= atom[N2].position.x;
    v.y -= atom[N2].position.y;
    v.z -= atom[N2].position.z;
    r1 = vector_length2(v);
    v = y;
    v.x -= twig[N0-n_atoms].x;
    v.y -= twig[N0-n_atoms].y;
    v.z -= twig[N0-n_atoms].z;
    r2 = vector_length2(v);
  } else if (INTERVAL(N2, n_atoms, n_atoms+n_twig) &&
             INTERVAL(N0, n0, n_atoms)) {
    x = atom[N1].position;
    x.x -= atom[N0].position.x;
    x.y -= atom[N0].position.y;
    x.z -= atom[N0].position.z;
    x = vector_scale(x, 2.038);
    x.x += atom[N0].position.x;
    x.y += atom[N0].position.y;
    x.z += atom[N0].position.z;
    y = twig[N3-n_atoms];
    y.x -= twig[N2-n_atoms].x;
    y.y -= twig[N2-n_atoms].y;
    y.z -= twig[N2-n_atoms].z;
    y = vector_scale(y, 2.038);
    y.x += twig[N2-n_atoms].x;
    y.y += twig[N2-n_atoms].y;
    y.z += twig[N2-n_atoms].z;
    v = x;
    v.x -= twig[N2-n_atoms].x;
    v.y -= twig[N2-n_atoms].y;
    v.z -= twig[N2-n_atoms].z;
    r1 = vector_length2(v);
    v = y;
    v.x -= atom[N0].position.x;
    v.y -= atom[N0].position.y;
    v.z -= atom[N0].position.z;
```

```
    r2 = vector_length2(v);
  }
  return(SCALE*(r1+r2)/BETA);
}
/* This routine returns the Coulomb, LJ, H-bond, and torsion
energies
   between the atoms in *atom and the atoms in *twig.
   The atoms in *twig must be those directly following those in
*atom.
   The atoms n_atoms to n_atoms+n_twig are in twig.
   The atoms n0 to n_atoms and n1 to n2 are in atom.
   n0 <= n_atoms <= n1 <= n2
*/
double delta_energy(torsion_list *t, hbond_list *l, atom_list
*atom,
                    vector *twig, int n_atoms, int n0, int n1, int
n2,
                    int n_twig)
{
  return(
         d_nonbond_energy(t, atom, twig, n_atoms, n0, n1, n2,
n_twig) +
         d_hbond_energy(l, atom, twig, n_atoms, n0, n1, n2, n_twig)
+
         d_torsion_energy(t, atom, twig, n_atoms, n0, n1, n2,
n_twig)
        );
}
/* This routine returns the total energy
*/
double energy(torsion_list *t, hbond_list *l, atom_list *atom,
              int n_atoms_total)
{
  return(
         nonbond_energy(t, atom, n_atoms_total) +
         hbond_energy(l, atom) +
         torsion_energy(t, atom)
        );
}
```

```
/* This routine returns the Coulomb and LJ energies
   between the atoms in *atom and the atoms in *twig.
   The atoms in *twig must be those directly following those in
*atom.
*/
double d_nonbond_energy(torsion_list *t, atom_list *atom, vector
*twig,
                        int n_atoms, int n0, int n1, int n2, int
n_twig)
{
#define FACT 332.06 /* converts from ei ej/rij to Kcal/mol */
  int i, j, k;
  vector r;
  double r2, r6, e, eij, rij, rij3, term, a, b;
  e = 0.0;
  for (i=n0; i<n2; i++) {
    if (INTERVAL(i,n_atoms,n1)) continue;
    for (j=0; j<n_twig; j++) {
      r.x = atom[i].position.x - twig[j].x;
      r.y = atom[i].position.y - twig[j].y;
      r.z = atom[i].position.z - twig[j].z;
      r2 = vector_length2(r);
      r6 = r2*r2*r2;
      eij = sqrt(atom[i].p->ei * atom[n_atoms+j].p->ei);
      rij = 0.5*(atom[i].p->ri + atom[n_atoms+j].p->ri);
      rij3 = rij*rij*rij;
      a = eij * rij3*rij3*rij3*rij3;
      b = 2*eij * rij3*rij3;
/* epsilon = 4*r */
      term = FACT * atom[i].p->charge * atom[n_atoms+j].p->charge
/ (4*r2)
             +    a/(r6*r6)  - b/r6;
      e += term;
    }
  }
/* subtract off 1/2 of 1-4 interactions */
  for (; t; t=t->next)
  {
    i = t->num[0]; j = t->num[3];
```

```
    if (INTERVAL(i,n_atoms,n_atoms+n_twig)) {
      k = i;
      i = j;
      j = k;
    }
     if    (INTERVAL(j,n_atoms,n_atoms+n_twig)    &&
(INTERVAL(i,n0,n_atoms) ||
       INTERVAL(i, n1, n2))) {
      r.x = atom[i].position.x - twig[j-n_atoms].x;
      r.y = atom[i].position.y - twig[j-n_atoms].y;
      r.z = atom[i].position.z - twig[j-n_atoms].z;
      r2 = vector_length2(r);
      r6 = r2*r2*r2;
      eij = sqrt(atom[i].p->ei * atom[j].p->ei);
      rij = 0.5 * (atom[i].p->ri + atom[j].p->ri);
      rij3 = rij*rij*rij;
      a = eij * rij3*rij3*rij3*rij3;
      b = 2*eij * rij3*rij3;
      term = FACT * atom[i].p->charge * atom[j].p->charge / (4*r2)
            + a/(r6*r6) - b/r6;
      e -= 0.5 * term;
    }
  }
  return(e);
#undef FACT
}
/* This routine returns the Coulomb and LJ energies
*/
double  nonbond_energy(torsion_list  *t,  atom_list  *atom,  int
n_atoms_total)
{
#define FACT 332.06 /* converts from ei ej/rij to Kcal/mol */
  int i, j;
  vector r;
  double r2, r6, e, eij, rij, rij3, term, a, b;
  e = 0.0;
  for (i=0; i<n_atoms_total; i++)
    for (j=i+1; j<n_atoms_total; j++) {
      r.x = atom[i].position.x - atom[j].position.x;
```

```
        r.y = atom[i].position.y - atom[j].position.y;
        r.z = atom[i].position.z - atom[j].position.z;
        r2 = vector_length2(r);
        r6 = r2*r2*r2;
        eij = sqrt(atom[i].p->ei * atom[j].p->ei);
        rij = 0.5*(atom[i].p->ri + atom[j].p->ri);
        rij3 = rij*rij*rij;
        a = eij * rij3*rij3*rij3*rij3;
        b = 2*eij * rij3*rij3;
/* epsilon = 4*r */
        term = FACT * atom[i].p->charge * atom[j].p->charge / (4*r2)
               +    a/(r6*r6)   - b/r6;
        e += term;
      }
/* subtract off 1/2 of 1-4 interactions */
  for (; t; t=t->next)
  {
    i = t->num[0]; j = t->num[3];
    r.x = atom[i].position.x - atom[j].position.x;
    r.y = atom[i].position.y - atom[j].position.y;
    r.z = atom[i].position.z - atom[j].position.z;
    r2 = vector_length2(r);
    r6 = r2*r2*r2;
    eij = sqrt(atom[i].p->ei * atom[j].p->ei);
    rij = 0.5 * (atom[i].p->ri + atom[j].p->ri);
    rij3 = rij*rij*rij;
    a = eij * rij3*rij3*rij3*rij3;
    b = 2*eij * rij3*rij3;
    term = FACT * atom[i].p->charge * atom[j].p->charge / (4*r2)
           + a/(r6*r6) - b/r6;
    e -= 0.5 * term;
  }
  return(e);
#undef fact
}
/* This routine returns the H-bond energy
   between the atoms in *atom and the atoms in *twig.
   The atoms in *twig must be those directly following those in
*atom.
```

```
*/
double d_hbond_energy(hbond_list *l, atom_list *atom, vector *twig,
                      int n_atoms, int n0, int n1, int n2, int
n_twig)
{
  int i,j,k;
  vector r;
  double r2, e;
  e = 0.0;
  for (; l; l=l->next) {
    i = l->num[0]; j = l->num[1];
    if (INTERVAL(i,n_atoms,n_atoms+n_twig)) {
      k = i;
      i = j;
      j = k;
    }
     if    (INTERVAL(j,n_atoms,n_atoms+n_twig)    &&
(INTERVAL(i,n0,n_atoms) ||
        INTERVAL(i,n1,n2))) {
      r.x = atom[i].position.x - twig[j-n_atoms].x;
      r.y = atom[i].position.y - twig[j-n_atoms].y;
      r.z = atom[i].position.z - twig[j-n_atoms].z;
      r2 = vector_length2(r);
       e    +=    l->p->a    /    (r2*r2*r2*r2*r2*r2)
l->p->b/(r2*r2*r2*r2*r2);
    }
  }
  return(e);
}
/* This routine returns the H-bond energy
*/
double hbond_energy(hbond_list *l, atom_list *atom)
{
  vector r;
  double r2, e;
  e = 0.0;
  for (; l; l=l->next) {
    r.x = atom[l->num[0]].position.x - atom[l->num[1]].position.x;
    r.y = atom[l->num[0]].position.y - atom[l->num[1]].position.y;
```

```
    r.z = atom[l->num[0]].position.z - atom[l->num[1]].position.z;
    r2 = vector_length2(r);
    e += l->p->a / (r2*r2*r2*r2*r2*r2) - l->p->b/(r2*r2*r2*r2*r2);
  }
  return(e);
}
/* This routine returns the H-bond energy
   between the atoms in *atom and the atoms in *twig.
   The atoms in *twig must be those directly following those in
*atom.
*/
double d_torsion_energy(torsion_list *t, atom_list *atom, vector
*twig,
                        int n_atoms, int n0, int n1, int n2, int
n_twig)
{
  int i,j,k,l;
  vector v[4];
  double theta, e, tmp;
  e = 0.0;
  for (; t; t=t->next)
  {
    if (t->p->v0[0] != 0.0 || t->p->v0[1] != 0.0 || t->p->v0[2] !=
0.0) {
      i = t->num[0]; j = t->num[1]; k = t->num[2]; l = t->num[3];
      if (INTERVAL(i,n_atoms+n_twig,n1) || i >= n2 || i < n0)
continue;
      if (INTERVAL(j,n_atoms+n_twig,n1) || j >= n2 || j < n0)
continue;
      if (INTERVAL(k,n_atoms+n_twig,n1) || k >= n2 || k < n0)
continue;
      if (INTERVAL(l,n_atoms+n_twig,n1) || l >= n2 || l < n0)
continue;
      if (!(INTERVAL(i,n_atoms,n_atoms+n_twig) ||
            INTERVAL(j,n_atoms,n_atoms+n_twig) ||
            INTERVAL(k,n_atoms,n_atoms+n_twig) ||
            INTERVAL(l,n_atoms,n_atoms+n_twig))) continue;
/*      printf("%d %d %d %d", i, j, k, l); */
      if (INTERVAL(i,n_atoms,n_atoms+n_twig))
```

```
        v[0] = twig[i-n_atoms]; else v[0] = atom[i].position;
      if (INTERVAL(j,n_atoms,n_atoms+n_twig))
        v[1] = twig[j-n_atoms]; else v[1] = atom[j].position;
      if (INTERVAL(k,n_atoms,n_atoms+n_twig))
        v[2] = twig[k-n_atoms]; else v[2] = atom[k].position;
      if (INTERVAL(l,n_atoms,n_atoms+n_twig))
        v[3] = twig[l-n_atoms]; else v[3] = atom[l].position;
      theta = torsion(v[0], v[1], v[2], v[3]);
      tmp = (t->p->v0[0]*(1 + cos(  theta-t->p->phi0[0])) +
            t->p->v0[1]*(1 + cos(2*theta-t->p->phi0[1])) +
            t->p->v0[2]*(1   +   cos(3*theta-t->p->phi0[2])))   /
t->degen;
/*      printf(" %lf %lf\n",theta,tmp); */
      e += tmp;
    }
  }
  return(e);
}
/* This routine returns the torsional energy
*/
double torsion_energy(torsion_list *t, atom_list *atom)
{
  double theta, e, tmp;
  e = 0.0;
  for (; t; t=t->next)
  {
    if (t->p->v0[0] != 0.0 || t->p->v0[1] != 0.0 || t->p->v0[2] !=
0.0) {
       theta    =    torsion(atom[t->num[0]].position,
atom[t->num[1]].position,
                              atom[t->num[2]].position,
atom[t->num[3]].position);
      tmp = (t->p->v0[0]*(1 + cos(  theta-t->p->phi0[0])) +
            t->p->v0[1]*(1 + cos(2*theta-t->p->phi0[1])) +
            t->p->v0[2]*(1   +   cos(3*theta-t->p->phi0[2])))   /
t->degen;
/*       printf("%d %d %d %d %lf %lf\n", t->num[0], t->num[1],
t->num[2],
                    t->num[3], theta, tmp); */
```

```
        e += tmp;
      }
    }
    return(e);
}

*************************************************************************
                MONTE CARLO ROUTINES - PEPTIDE5.C
*************************************************************************

/*                          The Monte Carlo routines
*/
#include "peptide.h"
/* This routine drives the configurational bias Monte Carlo
*/
void do_mc(rigid_unit *unit, torsion_list *t, hbond_list *l,
           atom_list *atom, atom_list *atom2, atom_info *atom_tmp,
           vector *twig[], regrowth *main, regrowth *side,
           int n_amino_acids, int n_atoms_total, int n_main, int
n_side,
           logical cyclic)
{
  int list_num, i, j;
  double logrosen, e, e2, emin;
  vector p0, b0;
  vector v1,v2;
  emin = 1.0E99;
  list_num = 0;
  p0.x = 0.0; p0.y = 0.0; p0.z = 0.0;
  b0.x = 0.0; b0.y = 0.0; b0.z = 1.0;
  e = 0;
  logrosen = 0;
/* create initial geomeotry */
  do_unit(&list_num, 0, n_atoms_total, n_atoms_total,
          &logrosen, unit, unit, t, l, atom, twig,
          p0, b0, &e);
/* read in initial geometry */
  if (0) read_restart(atom, n_atoms_total);
  if (cyclic)
```

```
    read_cycle(t, l, atom, main, side, twig, n_main, n_side,
n_atoms_total);
/*
  do_backbone_f(0, n_main, n_atoms_total, &logrosen, main,
                side, t, l, atom, twig, &e, TRUE);
  do_backbone_b(n_main-1, n_main, n_atoms_total, &logrosen, main,
                side, t, l, atom, twig, &e, TRUE);
  do_backbone_f_rigid(0, n_main, n_atoms_total,
                      &logrosen, main,
                      side, t, l, atom, atom_tmp, twig, &e, TRUE);
  do_backbone_b_rigid(n_main-1, n_main, n_atoms_total,
                      &logrosen, main,
                      side, t, l, atom, atom_tmp, twig, &e, TRUE);
*/
  emin = e = energy(t, l, atom, n_atoms_total);
/* copy old positions into new */
  for (j=0; j<n_atoms_total; j++) atom2[j] = atom[j];
/* do Monte Carlo */
  for (i=0; i<16000; i++) {
    printf("%d\n",i);
    rotate_main(atom, atom2, twig, main, side, t, l, n_main,
    n_atoms_total, &e);
/*
    regrow_main(t, l, atom, atom2, atom_tmp, twig, main, side,
                n_main, n_atoms_total, &e);
    regrow_side(t, l, atom, atom2, twig, main, side,
                n_side, n_atoms_total, &e);
*/
    if (e < emin) {
      emin = e;
      write_car_file(n_amino_acids,    n_atoms_total,    atom,
"min.car");
    }
  }
  printf("emin %lf\n",emin);
}
/* This routine reads in a restart file
*/
void read_restart(atom_list *atom, int n_atoms_total)
```

```
{
#define LINELEN 200
  FILE *fp;
  int i;
  char name[30], line[LINELEN];
  strcpy(name, "restart.car");
  if ((fp = fopen(name, "r")) == NULL) {
    printf("Data file %s does not exist\n", name);
    exit(1);
  }
  fgets(line, LINELEN, fp);
  fgets(line, LINELEN, fp);
  fgets(line, LINELEN, fp);
  fgets(line, LINELEN, fp);
  for (i=0; i<n_atoms_total; i++) {
    fgets(line, LINELEN, fp);
    sscanf(line, "%s %lf %lf %lf", name,
                                    &atom[i].position.x,
                                    &atom[i].position.y,
                                    &atom[i].position.z);
  }
  fclose(fp);
}
/* This routine reads in the backbone units plus one side-chain
atom
   for the geometry CXXXXXXC.  It then adds on each of the side
   groups randomly
*/
void read_cycle(torsion_list *t, hbond_list *l,
                atom_list *atom, regrowth *main, regrowth *side,
                vector  *twig[],  int  n_main,  int  n_side,  int
n_atoms_total)
{
#define LINELEN 200
  FILE *fp;
  int i, j, k, list_num;
  char name[30], line[LINELEN];
  double logrosen, e;
/* read in loop atoms plus one side group atom */
```

```
  if (n_main != 2*8+3) {
    printf("This cyclic geometry is not supported\n");
    exit(1);
  }
  strcpy(name, "CX6C.car");
  if ((fp = fopen(name, "r")) == NULL) {
    printf("Data file %s does not exist\n", name);
    exit(1);
  }
  fgets(line, LINELEN, fp);
  fgets(line, LINELEN, fp);
  fgets(line, LINELEN, fp);
  fgets(line, LINELEN, fp);
  for (i=0; i<n_main; i++) {
    /* printf("%d\n",main[i].unit->list_num); */
    for (j=0; j<main[i].unit->n_atoms; j++) {
      k = main[i].unit->list_num + j;
      fgets(line, LINELEN, fp);
      sscanf(line, "%s %lf %lf %lf", name,
                                        &atom[k].position.x,
                                        &atom[k].position.y,
                                        &atom[k].position.z);
      /* printf("%d %s %lf %lf %lf\n",k,name,
                                        atom[k].position.x,
                                        atom[k].position.y,
                                        atom[k].position.z); */
    }
    if (main[i].unit->n_bonds == 2) {
      k++;
      fgets(line, LINELEN, fp);
      sscanf(line, "%s %lf %lf %lf", name, &atom[k].position.x,
                                        &atom[k].position.y,
                                        &atom[k].position.z);
      /* printf("%d %s %lf %lf %lf\n",k,name,
                                        atom[k].position.x,
                                        atom[k].position.y,
                                        atom[k].position.z); */
    }
  }
```

```
  fclose(fp);
/* add on side groups */
  for (i=0; i<n_side; i++) {
    list_num = side[i].unit->list_num;
    do_unit(&list_num, 0, n_atoms_total, n_atoms_total,
          &logrosen, side[i].unit, side[i].unit, t, l, atom, twig,
          get_side_p0(atom, side, i), get_side_b0(atom, side, i),
          &e);
  }
}
/* This routine regrows from a main chain unit onwards
*/
void regrow_main(torsion_list *t, hbond_list *l,
                 atom_list *atom, atom_list *atom2,
                 atom_info *atom_tmp, vector *twig[],
                 regrowth *main, regrowth *side,
                 int n_main, int n_atoms_total, double *e)
{
  logical forward;
  int list_num, i, j, k;
  double logrosen1, logrosen2, x, e2, e1;
/* pick main group to start regrowth from */
  i = n_main*ran2(1.0);
/* pick direction to regrow */
  forward = (ran2(1.0) > 0.5);
  printf("regrowing %s from unit %d\n",(forward) ? "forward" :
"backward", i);
  list_num = main[i].unit->list_num;
/* copy old positions into new */
  for (j=0; j<n_atoms_total; j++) atom2[j].position =
atom[j].position;
/* regrow new peptide */
  e2 = 0;
  logrosen2 = 0.0;
  if (forward)
    do_backbone_f_rigid(i, n_main, n_atoms_total, &logrosen2, main,
                        side, t, l, atom2, atom_tmp, twig, &e2,
TRUE);
  else
```

```
    do_backbone_b_rigid(i, n_main, n_atoms_total, &logrosen2, main,
                        side, t, l, atom2, atom_tmp, twig, &e2,
TRUE);
  e2 = energy(t, l, atom2, n_atoms_total);
/* get old Rosenbluth weight */
  list_num = main[i].unit->list_num;
  e1 = 0.0;
  logrosen1 = 0.0;
  if (forward)
    do_backbone_f_rigid(i, n_main, n_atoms_total, &logrosen1, main,
                        side, t, l, atom, atom_tmp, twig, &e1,
FALSE);
  else
    do_backbone_b_rigid(i, n_main, n_atoms_total, &logrosen1, main,
                        side, t, l, atom, atom_tmp, twig, &e1,
FALSE);
  printf("Wn Wo %lf %lf\n",logrosen2, logrosen1);
  printf("En Eo %lf %lf\n",e2, *e);
/* perform acceptance test */
  x = 1.0;
  if (logrosen1 > logrosen2) x = exp(logrosen2-logrosen1);
/* accept new configuration */
  if (ran2(1.0) < x) {
    for (j=0; j<n_atoms_total; j++) atom[j].position =
atom2[j].position;
    *e = e2;
    printf("SWAP\n");
  }
}
/* This routine regrows a side chain
*/
void regrow_side(torsion_list *t, hbond_list *l,
                 atom_list *atom, atom_list *atom2, vector *twig[],
                 regrowth *main, regrowth *side,
                 int n_side, int n_atoms_total, double *e)
{
  int list_num, i, j, k, n1;
  double logrosen1, logrosen2, x, e2;
  if (n_side ==0 ) return;
```

```
/* pick main group to start regrowth from */
  i = n_side*ran2(1.0);
  printf("regrowing side chain %d\n",i);
  list_num = side[i].unit->list_num;
  logrosen2 = 0.0;
/* copy old positions into new */
  for (j=0; j<n_atoms_total; j++) atom2[j].position =
atom[j].position;
/* regrow side chain */
  e2 = 0;
/* determine n1 */
                                n          1                    =
side[i].prev->bond[side[i].prev->n_bonds-1]->next->list_num;
  do_unit(&list_num, 0, n1, n_atoms_total,
          &logrosen2, side[i].unit, side[i].unit, t, 1, atom2,
twig,
          get_side_p0(atom, side, i), get_side_b0(atom, side, i),
          &e2);
  e2 = energy(t, 1, atom2, n_atoms_total);
/* get old Rosenbluth weight */
  list_num = side[i].unit->list_num;
  logrosen1 = 0.0;
  old_unit(&list_num, 0, n1, n_atoms_total, &logrosen1,
           side[i].unit, side[i].unit, t, 1, atom, twig,
           get_side_p0(atom, side, i), get_side_b0(atom, side, i));
  printf("Wn Wo %lf %lf\n",logrosen2, logrosen1);
  printf("En Eo %lf %lf\n",e2, *e);
/* perform acceptance test */
  x = 1.0;
  if (logrosen1 > logrosen2) x = exp(logrosen2-logrosen1);
/* accept new configuration */
  if (ran2(1.0) < x) {
    for (j=side[i].unit->list_num; j<list_num; j++)
      atom[j].position = atom2[j].position;
    *e = e2;
    printf("SWAP\n");
  }
}
```

```
*************************************************************************
            CONCERTED ROTATION ROUTINES - PEPTIDE6.C
*************************************************************************

/*                          The concerted rotation routines
*/
#include "peptide.h"
/* global variables */
vector l[8], r[8];
double theta[8], m[3][3];
logical head[8];
/* This routine performs a concerted rotation on part of the main
chain.
*/
void rotate_main(atom_list *atom, atom_list *atom2, vector
                 *twig[], regrowth *main, regrowth *side,
                 torsion_list *t, hbond_list *l, int n_main, int
                 n_atoms_total, double *e)
{
  double jo, jn, logroseno, logrosenn, x, phi1, eo, en;
  int no, nn, i, j, i1, i2, i0;
  vector q;

  logical valid[4];
  double phi2[4], phi3[4], phi4[4], f[4];

  i0 = n_main * ran2(1.0);
  printf("Rotating from position %d\n",i0);
/* copy atom positions to atom2 */
  for   (i=0;   i<n_atoms_total;   i++)   atom2[i].position   =
atom[i].position;
/* determine theta, r, l */
  get_rot_params(atom, main, i0, n_main);
/* get original jacobian */
  jo = jac(atom, main, i0, n_main);
/* get constants needed by F5 */
  F5init(get_main_b0(atom, main, (i0+1) % n_main), &phi1);
/* get original Rosenbluth weight */
  eo = energy(t, l, atom, n_atoms_total);
```

```
  get_rot_rosenbluth(atom, atom2, twig, main, t, l, i0, n_main,
                     n_atoms_total, &no, &j, &logroseno, &en);
  printf("%d\n",no);
  if (no == 0) return; /* should never happen */
/* rotate r1 and get new constants */
  q = rotate_r1(atom, main, i0, n_main);
  F5init(q, &phi1);
/* get new Rosenbluth weight */
  get_rot_rosenbluth(atom, atom2, twig, main, t, l, i0, n_main,
                     n_atoms_total, &nn, &j, &logrosenn, &en);
  printf("%d\n",nn);
  if (nn == 0) return; /* geometric failure */
/* copy atomic positions */
  i1 = main[i0].unit->list_num;
  i2 = main[(i0+7) % n_main].unit->list_num;
  if (i2 < i1) i2 += n_atoms_total;
  for (i=i1; i<i2; i++)
    atom2[i % n_atoms_total].position = twig[j][i % n_atoms_total];
/* determine new Jacobian */
  jn = jac(atom2, main, i0, n_main);
/* Doros move */
  /* x = exp(-BETA*(en-eo)) * jn/jo * nn/no; */
/* CBMC move */
  if (logrosenn - logroseno < -10.0)
    x = 0.0;
  else if (logrosenn - logroseno > 10.0)
    x = 1.0;
  else
    x = jn/jo * exp(logrosenn - logroseno);
/* decide if move is accepted */
  printf("Wn Wo %lf %lf\n",logrosenn, logroseno);
  printf("En Eo %lf %lf\n",en, eo);
  if (ran2(1.0) < x) {
    printf("SWAP\n");
    *e = en;
/* copy atomic positions */
    i1 = main[i0].unit->list_num;
    i2 = main[(i0+7) % n_main].unit->list_num;
    if (i2 < i1) i2 += n_atoms_total;
```

```
    for (i=i1; i<i2; i++)
      atom[i % n_atoms_total].position = twig[j][i %
n_atoms_total];
    } else
    *e = eo;
}
/* This routine gets the theta, r, and l parameters */
void get_rot_params(atom_list *atom, regrowth *main, int i0,
                    int n_main)
{
  int i;
  vector t, v, v2;
  double len;
  rigid_unit *unit, *unit2, *unit3;
/* determine theta */
  for (i=0; i<8; i++) {
    unit = main[(i+i0) % n_main].unit;
    theta[i] = vector_dot(unit->head.axis,
                          unit->bond[unit->n_bonds-1]->tail.axis)
/
                          vector_length(unit->head.axis);
    theta[i] = (theta[i] < 1.0-EPS) ? acos(theta[i]) : 0.0;    }
/* determine r */
  for (i=0; i<8; i++) head[i] = TRUE;
  if (fabs(theta[5]) < EPS) head[5] = FALSE;
  for (i=0; i<8; i++) {
    unit = main[(i+i0) % n_main].unit;
    r[i] = atom[unit->list_num + ((head[i]) ? unit->head.atom_num
:
          unit->bond[unit->n_bonds-1]->tail.atom_num)].position;
  }
/* determine l */
  for (i=1; i<8; i++) {
    t.x = r[i].x - r[i-1].x;
    t.y = r[i].y - r[i-1].y;
    t.z = r[i].z - r[i-1].z;
    len = vector_length(t);
    /* if (2.03<len && len <2.05) len = 2.038;
    t = vector_scale(t, len); */
```

```
    l[i].x = len; l[i].y = l[i].z = 0.0;
    if (((main[(i+i0) % n_main].prev->type == Cunit) &&
         head[i-1]) || !head[i]) {
      l[i].x = vector_dot(t, get_main_b0(atom, main, (i+i0) %
n_main));
      l[i].y = sqrt(len * len - l[i].x * l[i].x);
    }
  }
/*
  for (i=1; i<8; i++) printf("%d %lf %lf %lf %lf\n",i, theta[i],
                             l[i].x, l[i].y, l[i].z);
  for (i=1; i<8; i++)
    printf("%d %lf %lf %lf\n",i, r[i].x, r[i].y, r[i].z);
*/
}
/* This routine checks the rigid unit theta values
*/
void check_theta(atom_list *atom, regrowth *main, int n_main)
{
  int i;
  vector t, v, v2, r;
  double len, theta;
  rigid_unit *unit, *unit2, *unit3;
  for (i=0; i<n_main; i++) {
    unit = main[i % n_main].unit;
    unit2 = main[i % n_main].prev;
    unit3 = main[(i+1) % n_main].unit;
    r = atom[unit->list_num + unit->head.atom_num].position;
    t = atom[unit2->list_num +

unit2->bond[unit2->n_bonds-1]->tail.atom_num].position;
    t.x = r.x - t.x; t.y = r.y - t.y; t.z = r.z - t.z;
      printf("%lf   %lf   ", vector_length(t),
vector_length(unit->head.axis));
    v = atom[unit3->list_num + unit3->head.atom_num].position;
    v2 = atom[unit->list_num +
         unit->bond[unit->n_bonds-1]->tail.atom_num].position;
    v.x -= v2.x; v.y -= v2.y; v.z -= v2.z;
    theta = vector_dot(t, v) / (vector_length(v)*vector_length(t));
```

```
      theta = (theta < 1.0-EPS) ? acos(theta) : 0.0;
      printf("%d %lf ",i, theta);
      theta = vector_dot(unit->head.axis,
                         unit->bond[unit->n_bonds-1]->tail.axis) /
                         vector_length(unit->head.axis);
      theta = (theta < 1.0-EPS) ? acos(theta) : 0.0;
      printf("%lf \n",theta);
    }
  }
  /* This routine determines the Rosenbluth weight */
  void get_rot_rosenbluth(atom_list *atom, atom_list *atom2,
                          vector *twig[], regrowth *main,
                          torsion_list *t, hbond_list *l, int i0,
                          int n_main, int n_atoms_total, int *n,
                          int *j, double *logrosen, double *e)
  {
    double phi[MAX_ROOTS][5], phi1, max, sum, de[MAX_ROOTS], ftmp;
    int i, k, k1, k2;
  /* get phi0-phi1 solutions */
    get_phi1(phi, n);
    if (*n == 0) return;
    if (*n > MAX_ROOTS) {
      printf("too many roots\n");
      *n = 0;
      return;
    }
  /* determine energies of solutions */
    max = -1E99;
    for (i=0; i<*n; i++) {
      get_r(phi[i][1], phi[i][2], phi[i][3], phi[i][4]);
      do_rotation(atom, twig[i], main, i0, n_main, n_atoms_total);
      k1 = main[i0].unit->list_num;
      k2 = main[(i0+7) % n_main].unit->list_num;
      if (k2 < k1) k2 += n_atoms_total;
      for (k=k1; k<k2; k++)
        atom2[k % n_atoms_total].position = twig[i][k %
  n_atoms_total];
      de[i] = -BETA*energy(t, l, atom2, n_atoms_total);
      if (de[i] > max) max = de[i];
```

```
    }
    sum = 0.0;
    for (i=0; i<*n; i++) {
      de[i] = exp(de[i] - max);
      sum += de[i];
    }
    *logrosen = log(sum) + max;
/* pick winner */
/* Doros move */
    /* *j = *n*ran2(1.0); */
/* CBMC move */
    de[0] /= sum;
    for (i=1; i<*n; i++) de[i] = de[i-1] + de[i]/sum;
    ftmp = ran2(1.0);
    for (*j=0; *j<*n; (*j)++) if (ftmp <= de[*j]) break;
/* get energy of winner */
    ftmp = de[*j];
    if (*j > 0) ftmp -= de[*j-1];
    ftmp *= sum;
    *e = -(log(ftmp)+max)/BETA;
/* assign r to the winner */
    get_r(phi[*j][1], phi[*j][2], phi[*j][3], phi[*j][4]);
}
/* This routine calculates the jacobian
*/
double jac(atom_list *atom, regrowth *main, int i0, int n_main)
{
    int i;
    vector u[7], h[6], t, v;
    double b[5][5];

/* form ui and hi */
    for (i=1; i<7; i++) u[i] = get_main_b0(atom, main, (i0+i;
%n_main);
    for (i=1; i<5; i++) h[i] = r[i];
    h[5] = atom[main[(i0+5)%n_main].unit->list_num +
                main[(i0+5)%n_main].unit->head.atom_num].position;
    v.x = r[6].x - h[5].x; v.y = r[6].y - h[5].y;
    v.z = r[6].z - h[5].z;
```

```
    v = vector_scale(v, 1.0);
  /* form B matrix */
    for (i=1; i<6; i++) {
      t.x = r[5].x - h[i].x;
      t.y = r[5].y - h[i].y;
      t.z = r[5].z - h[i].z;
      t = vector_cross(u[i], t);
      b[0][i-1] = t.x;
      b[1][i-1] = t.y;
      b[2][i-1] = t.z;
    }
    for (i=1; i<6; i++) {
      t = vector_cross(u[i], u[6]);
      b[3][i-1] = t.x;
      b[4][i-1] = t.y;
    }
    return(1.0/fabs(det5(b)));
  }
  /* This routine rotates phi0 to change r[1].
     It returns the new b0 for unit i0+1.
  */
  vector rotate_r1(atom_list *atom, regrowth *main, int i0, int
                   n_main)
  {
    double c, s, y;
    vector x, n;
  /* choose delta phi0 */
    y = DPHI * (1-2*ran2(1.0));
    c = cos(y);
    s = sin(y);
    n = get_main_b0(atom, main, i0);
  /* rotate about axis */
    x = r[1];
    x.x -= r[0].x;
    x.y -= r[0].y;
    x.z -= r[0].z;
    x = vector_rotate(x, n, c, s);
    r[1].x = r[0].x + x.x;
    r[1].y = r[0].y + x.y;
```

```
  r[1].z = r[0].z + x.z;
/* compute new b0 for unit i0+1 */
  return(vector_rotate(get_main_b0(atom, main, (i0+1) % n_main),
n, c, s));
}
/* This routine constructs r2-r4 from the theta, phi
information  */
void get_r(double phi1, double phi2, double phi3, double phi4)
{
  int i;
  vector x, y;
/*
  printf("\n");
  printf("%lf %lf %lf %lf %lf\n", phi1, phi2, phi3, phi4);
*/
  x = bxm(m, l[1]);
  r[1].x = x.x + r[0].x;
  r[1].y = x.y + r[0].y;
  r[1].z = x.z + r[0].z;
  x = bxm(m, flory_rot(theta[1], phi1, l[2]));
  r[2].x = x.x + r[1].x;
  r[2].y = x.y + r[1].y;
  r[2].z = x.z + r[1].z;
  x = bxm(m, flory_rot(theta[1], phi1, flory_rot(theta[2], phi2,
l[3])));
  r[3].x = x.x + r[2].x;
  r[3].y = x.y + r[2].y;
  r[3].z = x.z + r[2].z;
  x = bxm(m, flory_rot(theta[1], phi1, flory_rot(theta[2],
          phi2, flory_rot(theta[3], phi3, l[4]))));
  r[4].x = x.x + r[3].x;
  r[4].y = x.y + r[3].y;
  r[4].z = x.z + r[3].z;
/*
  for (i=1; i<7; i++)
     printf("%d %lf %lf %lf\n",i, r[i].x, r[i].y, r[i].z);
*/
}
/* This routine rotates the rigid units to the positions
```

```
    of the concerted rotation.
 */
 void do_rotation(atom_list *atom, vector *twig, regrowth *main,
                  int i0, int n_main, int n_atoms_total)
 {
   int i, j, i1, i2, i3, j2;
   double m[3][3], a[3][3], tmp, len2;
   vector x1, x2, y1, y2, x;
   rigid_unit *unit;
   for (i=-1; i<6; i++) {
   i1 = (i+i0+n_main) % n_main;
   i2 = (i+i0+1) % n_main;
   i3 = (i+i0+2) % n_main;
/* get x1 & x2 */
   x1 = r[i+1];
   x = (i > -1) ?

twig[main[i1].unit->bond[main[i1].unit->n_bonds-1]->tail.atom_num+
             main[i1].unit->list_num] :

atom[main[i1].unit->bond[main[i1].unit->n_bonds-1]->tail.atom_num+
             main[i1].unit->list_num].position;
    x1.x -=  x.x;  x1.y -= x.y; x1.z -= x.z;
    x2 = atom[main[i2].unit->list_num + ((head[i+1]) ?
              main[i2].unit->head.atom_num :

main[i2].unit->bond[main[i2].unit->n_bonds-1]->tail.atom_num)]
              .position;
                                        x                    =
atom[main[i1].unit->bond[main[i1].unit->n_bonds-1]->tail.atom_num
+
             main[i1].unit->list_num].position;
    x2.x -= x.x; x2.y -= x.y; x2.z -= x.z;
/* get rotation matrix */
    flory_lab(a, x1, x2);
/* get y1 & y2 */
    y1 = r[i+2];
    x = (i > -1) ?
```

```
twig[main[i1].unit->bond[main[i1].unit->n_bonds-1]->tail.atom_num+
            main[i1].unit->list_num] :

atom[main[i1].unit->bond[main[i1].unit->n_bonds-1]->tail.atom_num+
            main[i1].unit->list_num].position;
    y1.x -= x.x; y1.y -= x.y; y1.z -= x.z;
    y2 = atom[main[i3].unit->list_num + ((head[i+2]) ?
             main[i3].unit->head.atom_num :

main[i3].unit->bond[main[i3].unit->n_bonds-1]->tail.atom_num)]
             .position;
                                          x                              =
atom[main[i1].unit->bond[main[i1].unit->n_bonds-1]->tail.atom_num
+
            main[i1].unit->list_num].position;
    y2.x -= x.x; y2.y -= x.y; y2.z -= x.z;
    y2 = mxb(a, y2);
/* get projection */
    len2 = vector_length2(x1);
    tmp = vector_dot(y2, x1) / len2;
    y2.x -= x1.x * tmp;
    y2.y -= x1.y * tmp;
    y2.z -= x1.z * tmp;
    tmp = vector_dot(y1, x1) / len2;
    y1.x -= x1.x * tmp;
    y1.y -= x1.y * tmp;
    y1.z -= x1.z * tmp;
/* get rotation matrix */
    flory_lab(m, y1, y2);
    mxm (m, a);
/* perform rotation */
                                    x       1                          =
atom[main[i1].unit->bond[main[i1].unit->n_bonds-1]->tail.atom_num+
             main[i1].unit->list_num].position;
    x2 = (i > -1) ?

twig[main[i1].unit->bond[main[i1].unit->n_bonds-1]->tail.atom_num+
          main[i1].unit->list_num] : x1;
    j2 = main[i3].unit->list_num;
```

```
    if (i3 == 0) j2 = n_atoms_total;
    for (j=main[i2].unit->list_num; j < j2; j++) {
      x = atom[j].position;
      x.x -= x1.x;
      x.y -= x1.y;
      x.z -= x1.z;
      x = mxb(m, x);
      x.x += x2.x;
      x.y += x2.y;
      x.z += x2.z;
      twig[j] = x;
    }
  }
}
/* This routine determines the phi1-phi3 values
*/
void get_phi1(double phi[MAX_ROOTS][5], int *n)
{
#define NTRY 10000
  int i, j;
  logical valid[NTRY+1][4];
  double phi1[NTRY+1], phi2[4], phi3[4], phi4[4];
  double f[NTRY+1][4];
  *n = 0;
  i = 0;
/* Evaluate F5 */
  for (i=0; i<=NTRY; i++) {
    phi1[i] = -PI + i*2*PI/NTRY;
    F5(phi1[i], phi2, phi3, phi4, f[i], valid[i]);
  }
/* Now search for roots */
  for (i=0; i<NTRY; i++) {
    for (j=0; j<4; j++) {
      if (!valid[i][j] || !valid[i+1][j]) continue;
      if ((f[i][j] < 0 && f[i+1][j] > 0) ||
          (f[i][j] > 0 && f[i+1][j] < 0)) {
        if (*n >= MAX_ROOTS) {
          printf("Excessive number of roots failure in
get_phi1\n");
```

```
            return;
          }
          get_root(phi1[i], phi1[i+1], &phi[*n][1], &phi[*n][2],
                   &phi[*n][3], &phi[*n][4], j);
          (*n)++;
        }
      }
    }
#undef NTRY
}
/* This routine refines a root using bisection
*/
void get_root(double x0, double x1, double *p1, double *p2,
double *p3,                 double *p4, int n)
{
  logical valid[4];
  double phi2[4], phi3[4], phi4[4], f[4];
/* order roots: f(x0) < 0 && f(x1) > 0 */
  F5(x1, phi2, phi3, phi4, f, valid);
  if (f[n] < 0.0) {
    *p1 = x0;
    x0 = x1;
    x1 = *p1;
  }
/* do bisection to refine root */
  do {
    *p1 = 0.5*(x1+x0);
    F5(*p1, phi2, phi3, phi4, f, valid);
    if (f[n] > 0) x1 = *p1; else x0 = *p1;
  } while (fabs(x1-x0) > EPS);
  *p2 = phi2[n];
  *p3 = phi3[n];
  *p4 = phi4[n];
}
/* constants */
double c10, c11, c12, q12, c20, c21, c22, fact1, fact2;
vector x0, u60;
/* This routine sets up constants that F5 uses.
   The constants are independent of phi1
```

```
*/
void F5init(vector q2, double *phi1)
{
  int i,j;
  vector t;
  double c1, c2, a[3][3], tmp;
  t.x = 1.0; t.y = t.z = 0.0;
  flory_labinv(m, q2, t);
  t.x = r[1].x - r[0].x; t.y = r[1].y - r[0].y; t.z = r[1].z -
r[0].z;
  t = mxb(m, t);
  if (fabs(t.y) < EPS && fabs(t.z) < EPS) {
    c1 = 1.0;
    c2 = 0.0;
  } else {
    c1 = (l[1].y*t.y + t.z*l[1].z)/(t.y*t.y + t.z*t.z);
    c2 = (-l[1].z*t.y + t.z*l[1].y)/(t.y*t.y + t.z*t.z);
    if (fabs(c1) < EPS && fabs(c2) < EPS) c1 = 1.0;
  }
  a[0][0] = 1; a[0][1] = 0; a[0][2] = 0;
  a[1][0] = 0; a[1][1] = c1; a[1][2] = c2;
  a[2][0] = 0; a[2][1] = -c2; a[2][2] = c1;
  mxm(a, m);
  for (i=0; i<3; i++)
    for (j=0; j<3; j++)
      m[i][j] = a[i][j];
  t.x = r[2].x - r[1].x; t.y = r[2].y - r[1].y; t.z = r[2].z -
r[1].z;
  t = mxb(m, t);
  tmp = (sin(theta[1])*l[2].x - cos(theta[1])*l[2].y);
  *phi1 = atan2(t.z/tmp, t.y/tmp);
  x0.x = r[5].x - r[0].x; x0.y = r[5].y - r[0].y; x0.z = r[5].z -
r[0].z;
  x0 = mxb(m, x0);
  x0.x -= l[1].x;
  x0.y -= l[1].y;
  x0.z -= l[1].z;
  if (fabs(theta[5]) < EPS && fabs(theta[3]) < EPS) {
    c10 = l[3].x*cos(theta[4]);
```

```
    c11 = -(cos(theta[2])*l[3].x + sin(theta[2])*l[3].y);
    tmp = sin(theta[2])*l[3].x - cos(theta[2])*l[3].y;
    c10 /= tmp;
    c11 /= tmp;
  } else if (fabs(theta[5]) < EPS && fabs(theta[3]) > EPS) {
    c10 = -l[5].x - l[4].x*cos(theta[4]);
    c11 = -(cos(theta[2])*l[3].x + sin(theta[2])*l[3].y);
    c12 = 1.0/(sin(theta[2])*l[3].x - cos(theta[2])*l[3].y);
  } else if (fabs(theta[3]) > EPS) {
    t.z = 0.0;
    t.x = l[4].x*cos(theta[4]) - l[4].y*sin(theta[4]) + l[5].x;
    t.y = l[4].x*sin(theta[4]) + l[4].y*cos(theta[4]) + l[5].y;
    q12 = vector_length2(t);
    c10 = q12 - vector_length2(l[3]);
    c11 = 2*(cos(theta[2])*l[3].x + sin(theta[2])*l[3].y);
    c12 = -1.0/(2*(sin(theta[2])*l[3].x - cos(theta[2])*l[3].y));
  } else {
    c10 = l[3].x + l[4].x + l[5].x*cos(theta[4]);
    c11 = -cos(theta[2]);
    tmp = sin(theta[2]);
    c10 /= tmp;
    c11 /= tmp;
  }
  c20 = vector_length2(l[5]) - vector_length2(l[4]);
  c21 = 2*(cos(theta[3])*l[4].x + sin(theta[3])*l[4].y);
  c22 = -1.0/(2*(sin(theta[3])*l[4].x - cos(theta[3])*l[4].y));
  fact1 = sin(theta[4])*l[5].x - cos(theta[4])*l[5].y;
  fact2 = l[6].x*cos(theta[5]) + l[6].y*sin(theta[5]);
  u60.x = r[6].x - r[5].x; u60.y = r[6].y - r[5].y; u60.z = r[6].z
- r[5].z;
}
/* This routine returns the F5 function of Doros.
   *n is the number of solutions, which are in f.
*/
void F5(double phi1, double phi2[4], double phi3[4], double
        phi4[4], double f[4], logical valid[4])
{
  int i, j;
  double tmp, c1, c2;
```

```
  vector v1, q1, q2, x, y, t, u6;
  double a[3][3], rot1[3][3], rot2[3][3], rot3[3][3], rot4[3][3];
/* determine c1 */
  valid[0] = valid[1] = valid[2] = valid[3] = FALSE;
  flory_rot_matrix(theta[1], phi1, rot1);
  x = bxm(rot1, x0);
  x.x -= l[2].x; x.y -= l[2].y; x.z -= l[2].z;
  v1 = x;
  if (fabs(theta[5]) < EPS && fabs(theta[3]) < EPS) {
    x = bxm(rot1, mxb(m, vector_scale(u60, 1.0)));
    c1 = (c10 + x.x*c11) / sqrt(x.y*x.y + x.z*x.z);
  } else if (fabs(theta[5]) < EPS && fabs(theta[3]) > EPS) {
    x = bxm(m, flory_rot(theta[1], phi1, l[2]));
    r[2].x = x.x + r[1].x; r[2].y = x.y + r[1].y; r[2].z = x.z +
r[1].z;
    t.x = r[5].x - r[2].x; t.y = r[5].y - r[2].y; t.z = r[5].z -
r[2].z;
    x = bxm(rot1, mxb(m, vector_scale(u60,1.0)));
    c1 = c12*(c10 +  vector_dot(t,
         u60)/vector_length(u60)  +  x.x*c11)  /  sqrt(x.y*x.y  +
x.z*x.z);
  } else if (fabs(theta[3]) > EPS) {
    c1 = c12*(c10 - vector_length2(x) + x.x*c11) / sqrt(x.y*x.y +
x.z*x.z);
  } else {
    c1 = (c10 + x.x*c11) / sqrt(x.y*x.y + x.z*x.z);
  }
  /* printf("c1 %lf\n",c1); */
  if (fabs(c1) > 1) return;
/* determine phi2 */
  tmp = asin(c1);
  phi2[0] = phi2[2] = -atan(x.y/x.z);
  if (x.z < 0) phi2[0] = phi2[2] = phi2[0] - PI;
  phi2[0] += tmp;
  phi2[2] += PI - tmp;
  phi2[1] = phi2[0];
  phi2[3] = phi2[2];
  x = v1;
/* determine c2 and phi3 */
```

```
  for (i=0; i<2; i++) {
    y = flory_rotinv(theta[2], phi2[2*i], x);
    y.x -= l[3].x; y.y -= l[3].y; y.z -= l[3].z;
    c2 = c22*(c20 - vector_length2(y) + y.x*c21) / sqrt(y.y*y.y +
y.z*y.z);
    /* printf("c2 %lf\n",c2); */
    if (fabs(c2) <= 1)  {
      tmp = asin(c2);
      phi3[2*i] = phi3[2*i+1] = -atan(y.y/y.z);
      if (y.z < 0) phi3[2*i] = phi3[2*i+1] = phi3[2*i+1] - PI;
      phi3[2*i] += tmp;
      phi3[2*i+1] += PI - tmp;
      valid[2*i] = valid[2*i+1] = TRUE;
    }
  }
  for (i=0; i<4; i++) {
    if (!valid[i]) continue;
/* determine r4 */
    flory_rot_matrix(theta[2], phi2[i], rot2);
    flory_rot_matrix(theta[3], phi3[i], rot3);
    x = mxb(rot3, l[4]);
    x.x += l[3].x; x.y += l[3].y; x.z += l[3].z;
    x = mxb(rot2, x);
    x.x += l[2].x; x.y += l[2].y; x.z += l[2].z;
    x = mxb(rot1, x);
    x.x += l[1].x; x.y += l[1].y; x.z += l[1].z;
    x = bxm(m, x);
    x.x += r[0].x; x.y += r[0].y; x.z += r[0].z;
/* determine F5 */
    if (fabs(theta[5]) < EPS && fabs(theta[3]) < EPS) {
      v1.x = r[6].x - x.x; v1.y = r[6].y - x.y; v1.z = r[6].z -
x.z;
      f[i] = sqrt((l[6].x+l[5].x)*(l[6].x+l[5].x) +
                  l[5].y*l[5].y) - vector_length(v1);
    } else if (fabs(theta[5]) < EPS && fabs(theta[3]) > EPS) {
      x = bxm(m, mxb(rot1, mxb(rot2, mxb(rot3, l[4]))));
      f[i] = vector_dot(x, u60) /
             (vector_length(x)*vector_length(u60)) - cos(theta[4]);
    } else {
```

```
      x.x = r[5].x - x.x; x.y = r[5].y - x.y; x.z = r[5].z - x.z;
      x = mxb(m, x);
      x = bxm(rot3, bxm(rot2, bxm(rot1, x)));
      phi4[i] = atan2(x.z/fact1, x.y/fact1);
      u6 = mxb(m, u60);
      x.x = 1.0; x.y = 0; x.z = 0;
      f[i] = vector_dot(u6, mxb(rot1, mxb(rot2, mxb(rot3,
                        flory_rot(theta[4],  phi4[i],  x))))) -
fact2;
    }
  }
}

*************************************************************************
            GEOMETRY/ROTATION ROUTINES - PEPTIDE7.C
*************************************************************************

/*                        The geometry routines
*/
#include "peptide.h"
/*  This  routine  rotates  the  vector  a  about  n  by  theta
(counterclockwise is +)
  r' = r cos(theta) + n(n.r)(1-cos(theta)) + nxr sin(theta)
*/
vector vector_rotate(vector a, vector n, double cos_theta, double
sin_theta)
{
  double fact;
  vector ret, v;
  fact = (n.x*a.x + n.y*a.y + n.z*a.z) * (1.0 - cos_theta);
  v = vector_cross(n,a);
  ret.x = a.x*cos_theta + n.x*fact + v.x*sin_theta;
  ret.y = a.y*cos_theta + n.y*fact + v.y*sin_theta;
  ret.z = a.z*cos_theta + n.z*fact + v.z*sin_theta;
  return(ret);
}
/* This routine returns main-chain b0
   i=0 noncyclic case should never happen--it won't be right
*/
```

```
vector get_main_b0(atom_list *atom, regrowth *main, int i)
{
  vector x, y;
  if (main[i].prev == NULL) {
    x.x = x.y = 0.0;
    x.z = 1.0;
    return(x);
  }
   x    =    atom[main[i].unit->list_num    +
main[i].unit->head.atom_num].position;
                                                          y    =
atom[main[i].prev->bond[main[i].prev->n_bonds-1]->tail.atom_num +
          main[i].prev->list_num].position;
  x.x -= y.x;
  x.y -= y.y;
  x.z -= y.z;
  return(vector_scale(x, 1.0));
}
/* This routine returns main-chain p0
   i=0 noncyclic case should never happen--it won't be right
*/
vector get_main_p0(atom_list *atom, regrowth *main, int i)
{
  vector x;
  if (main[i].prev == NULL) {
    x.x = x.y = x.z = 0.0;
    return(x);
  }
                                                          x    =
atom[main[i].prev->bond[main[i].prev->n_bonds-1]->tail.atom_num +
          main[i].prev->list_num].position;
  return(x);
}
/* This routine returns side-chain b0 */
vector get_side_b0(atom_list *atom, regrowth *side, int i)
{
  vector x, y;
   x    =    atom[side[i].unit->list_num    +
side[i].unit->head.atom_num].position;
```

```
    y    =    atom[side[i].prev->list_num    +
side[i].prev->head.atom_num].position;
  x.x -= y.x;
  x.y -= y.y;
  x.z -= y.z;
  return(vector_scale(x, 1.0));
}
/* This routine returns side-chain p0 */
vector get_side_p0(atom_list *atom, regrowth *side, int i)
{
  vector x;
    x    =    atom[side[i].prev->list_num    +
side[i].prev->head.atom_num].position;
  return(x);
}
/* This routine gives the Flory rotation matrix
*/
void flory_rot_matrix(double theta, double phi, double m[3][3])
{
  double cost, sint, cosp, sinp;
  cost = cos(theta); sint = sin(theta);
  cosp = cos(phi); sinp = sin(phi);
  m[0][0] = cost;
  m[0][1] = sint;
  m[0][2] = 0.0;
  m[1][0] = sint*cosp;
  m[1][1] = -cost*cosp;
  m[1][2] = sinp;
  m[2][0] = sint*sinp;
  m[2][1] = -cost*sinp;
  m[2][2] = -cosp;
}
/* This routine does the Flory rotation
*/
vector flory_rot(double theta, double phi, vector a)
{
  vector t;
  double cost, sint, cosp, sinp, tmp;
  cost = cos(theta); sint = sin(theta);
```

```
  cosp = cos(phi); sinp = sin(phi);
  tmp = sint*a.x - cost*a.y;
  t.x = cost*a.x + sint*a.y;
  t.y = cosp*tmp + sinp*a.z;
  t.z = sinp*tmp - cosp*a.z;
  return(t);
}
/* This routine does the inverse Flory rotation
*/
vector flory_rotinv(double theta, double phi, vector a)
{
  vector t;
  double cost, sint, cosp, sinp, tmp;
  cost = cos(theta); sint = sin(theta);
  cosp = cos(phi); sinp = sin(phi);
  tmp = cosp*a.y + sinp*a.z;
  t.x = cost*a.x + sint*tmp;
  t.y = sint*a.x - cost*tmp;
  t.z = sinp*a.y - cosp*a.z;
  return(t);
}
/* This routine constructs the lab transformation to go from l to
r
*/
void flory_lab(double m[3][3], vector r, vector l)
{
  double sin_theta, cos_theta;
  vector n;
  r = vector_scale(r, 1.0);
  l = vector_scale(l, 1.0);
  n = vector_cross(l,r);
  cos_theta = vector_dot(l,r);
  sin_theta = vector_length(n);
  if (sin_theta < EPS) {
    n.x = 1.0;
  } else {
    n.x /= sin_theta;
    n.y /= sin_theta;
    n.z /= sin_theta;
```

```
  }
  m[0][0] = cos_theta + n.x*n.x*(1.0-cos_theta)                ;
  m[0][1] =             n.x*n.y*(1.0-cos_theta) - sin_theta*n.z;
  m[0][2] =             n.x*n.z*(1.0-cos_theta) + sin_theta*n.y;
  m[1][0] =             n.y*n.x*(1.0-cos_theta) + sin_theta*n.z;
  m[1][1] = cos_theta + n.y*n.y*(1.0-cos_theta)                ;
  m[1][2] =             n.y*n.z*(1.0-cos_theta) - sin_theta*n.x;
  m[2][0] =             n.z*n.x*(1.0-cos_theta) - sin_theta*n.y;
  m[2][1] =             n.z*n.y*(1.0-cos_theta) + sin_theta*n.x;
  m[2][2] = cos_theta + n.z*n.z*(1.0-cos_theta)                ;
}
/* This routine constructs the inverse lab transformation
*/
void flory_labinv(double m[3][3], vector r, vector l)
{
  double sin_theta, cos_theta;
  vector n;
  r = vector_scale(r, 1.0);
  l = vector_scale(l, 1.0);
  n = vector_cross(l,r);
  cos_theta = vector_dot(l,r);
  sin_theta = vector_length(n);
  if (sin_theta < EPS) {
    n.x = 1.0;
  } else {
    n.x /= sin_theta;
    n.y /= sin_theta;
    n.z /= sin_theta;
  }
  m[0][0] = cos_theta + n.x*n.x*(1.0-cos_theta)                ;
  m[1][0] =             n.x*n.y*(1.0-cos_theta) - sin_theta*n.z;
  m[2][0] =             n.x*n.z*(1.0-cos_theta) + sin_theta*n.y;
  m[0][1] =             n.y*n.x*(1.0-cos_theta) + sin_theta*n.z;
  m[1][1] = cos_theta + n.y*n.y*(1.0-cos_theta)                ;
  m[2][1] =             n.y*n.z*(1.0-cos_theta) - sin_theta*n.x;
  m[0][2] =             n.z*n.x*(1.0-cos_theta) - sin_theta*n.y;
  m[1][2] =             n.z*n.y*(1.0-cos_theta) + sin_theta*n.x;
  m[2][2] = cos_theta + n.z*n.z*(1.0-cos_theta)                ;
}
```

```
/* This routine returns a vector cross product
*/
vector vector_cross(vector a, vector b)
{
  vector ret;
  ret.x = a.y*b.z - a.z*b.y;
  ret.y = a.z*b.x - a.x*b.z;
  ret.z = a.x*b.y - a.y*b.x;
  return(ret);
}
/* This function scales the vector v so that |v| = r
*/
vector vector_scale(vector v, double r)
{
  double ftmp;
  ftmp = sqrt(v.x*v.x + v.y*v.y + v.z*v.z);
  v.x *= r/ftmp;
  v.y *= r/ftmp;
  v.z *= r/ftmp;
  return(v);
}
/* This routine returns mxn in m
*/
void mxm(double m[3][3], double n[3][3])
{
  int i,j,k;
  double a[3][3];
  for (i=0; i<3; i++)
    for (j=0; j<3; j++) {
      a[i][j] = 0.0;
      for (k=0; k<3; k++) a[i][j] += m[i][k]*n[k][j];
    }
  for (i=0; i<3; i++)
    for (j=0; j<3; j++)
      m[i][j] = a[i][j];
}
/* This routine deturns det(m), where m is 5x5
*/
double det5(double m[5][5])
```

```
{
  int i,j,k;
  double a[5][5], fact;
  for (i=0; i<5; i++)
    for (j=0; j<5; j++)
      a[i][j] = m[i][j];
  for (i=0; i<4; i++) {
    for (k=i+1; k<5; k++) {
      fact = a[k][i] / a[i][i];
      for (j=i; j<5; j++) a[k][j] -= fact*a[i][j];
    }
  }
  return(a[0][0]*a[1][1]*a[2][2]*a[3][3]*a[4][4]);
}
/* This routine returns det(m), where m is 3x3
*/
double det(double m[3][3])
{
  return(m[0][0]*m[1][1]*m[2][2] + m[0][1]*m[1][2]*m[2][0] +
         m[0][2]*m[1][0]*m[2][1] - m[2][0]*m[1][1]*m[0][2] -
         m[1][0]*m[0][1]*m[2][2] - m[0][0]*m[2][1]*m[1][2]);
}
/* This routine returns Mb
*/
vector mxb(double m[3][3], vector b)
{
  vector t;

  t.x = m[0][0]*b.x + m[0][1]*b.y + m[0][2]*b.z;
  t.y = m[1][0]*b.x + m[1][1]*b.y + m[1][2]*b.z;
  t.z = m[2][0]*b.x + m[2][1]*b.y + m[2][2]*b.z;
  return(t);
}
/* This routine returns Mb
*/
vector bxm(double m[3][3], vector b)
{
  vector t;
```

```
  t.x = m[0][0]*b.x + m[1][0]*b.y + m[2][0]*b.z;
  t.y = m[0][1]*b.x + m[1][1]*b.y + m[2][1]*b.z;
  t.z = m[0][2]*b.x + m[1][2]*b.y + m[2][2]*b.z;
  return(t);
}
/* This routine returns b1.b2
*/
double vector_dot(vector b1, vector b2)
{
  return(b1.x*b2.x + b1.y*b2.y + b1.z*b2.z);
}
/* This routine returns |v|
*/
double vector_length(vector v)
{
  return(sqrt(v.x*v.x + v.y*v.y + v.z*v.z));
}
/* This routine returns |v|^2
*/
double vector_length2(vector v)
{
  return(v.x*v.x + v.y*v.y + v.z*v.z);
}

*************************************************************************
                RANDOM NUMBER GENERATOR - RANDOM.C
*************************************************************************

/*
  This is the pseudo-random number library.
*/
#include <time.h>
/*
  This function returns a random number in [0,1).
  It uses a linear-congruential method.
  ran(0.0) initializes the random number seed with a time dependant
value
    and returns the value of the seed that the generator
recognizes.
```

```
  ran(1.0) returns the next number in the random sequence.
  Other arguments initialize the seed with the user-supplied value.
    Initializing the generator with a seed from the sequence, will
cause the
    subsequent ran(1.0) to generate the next value of the sequence.
    This is usefull, for example, to shut down and start up the
generator
    without a loss of continuity in the sequence.
  Values r 1 or < 0 are not recommended.
  It has a period of M.
*/
double ran(double dummy)
{
  static long int ix;
  double rm = 566927.0, rm2 = 1.0/rm;
  long int k = 5701, j = 3621, m = 566927, tmp;
     /* make sure parameters not too far off */
  if (dummy > 2.0) dummy = 2.0;
  if (dummy < -2.0) dummy = -2.0;
  if (dummy != 1.0)
  {
      if ((tmp = dummy*rm) < m)
        ix = tmp;
      else
        ix = m-1;
      if (ix < 0)
        ix = 0;
  } else
    ix = (j*ix + k) % m;
  return(ix * rm2);
}
/*
  This function returns a pseudo-random number in (0,1).
  This is a more robust pseudo-random number generator than a
simple linear-
    congruential gererator is.
  It uses three linear congruential generators to get one random
number.
  ran2(0.0) initializes the generator with time-dependent values.
```

```
  ran2(1.0) returns a pseudo-random number.
  Other arguments are used as an initializing seed.
  Arguments r 1 or s 0 are ill-advised.
  It has a period of (m1-1)(m2-1)(m3-1)/4.
*/
double ran2(double dummy)
{
  double f1=1.0/30269.0 ,f2=1.0/30307.0, f3=1.0/30323.0, tmp;
  int m1=30269, m2=30307, m3=30323, seed, itmp;
  static x,y,z;
     /* make sure parameters not too far off */
  if (dummy > 1.1) dummy = 1.1;
  if (dummy < -1.1) dummy = -1.1;
  if (dummy != 1.0)
  {
                /* initialize with user's seed value */
      if ((itmp = dummy*m1) < m1)
        seed = itmp;
      else
        seed = m1-1;
    if (seed < 1) seed = 1;
                                /* initialize first generator */
    x = seed;
                                /* initialize second generator */
    y = 172 * (x % 176)  -  35 * (x/176);
    if (y < 0) y += m2;
                                /* initialize third generator */
    z = 170 * (y % 178)  -  63 * (y/178);
    if (z < 0) z += m3;
  }
                                        /* first generator */
  x = 171 * (x % 177)  -  2 * (x/177);
  if (x < 0) x += m1;
                                        /* second generator */
  y = 172 * (y % 176)  -  35 * (y/176);
  if (y < 0) y += m2;
                                        /* third generator */
  z = 170 * (z % 178)  -  63 * (z/178);
  if (z < 0) z += m3;
```

```
                                        /* amalgamated result */
  itmp = tmp = x*f1 + y*f2 + z*f3;
  return(tmp - itmp);
}

*************************************************************************
*************************************************************************
                          C INCLUDE FILES
*************************************************************************
*************************************************************************

*************************************************************************
              GLOBAL VARIABLE TYPES - PEP_TYPE.H
*************************************************************************

/* Global types used in the program */
typedef enum {FALSE, TRUE} logical;
typedef enum {BAD, G, A, V, L, I, S, T, D, E, N, Q, K, H, R, F, Y,
W, C, M, P}
      acid_label;
typedef enum {UNKNOWN, nonCunit, Cunit} unit_label;
typedef struct {
                double x,y,z;
              } vector;
typedef struct {
                vector axis;
                int atom_num;
                int bond[MAX_BONDS];
              } connector;
typedef struct bond_struct {
                connector tail;
                struct rigid_unit_struct *next;
              } bond_type;
typedef char *string;

typedef struct {
                char name[NAME_LENGTH];
                char type[NAME_LENGTH];
                double charge, ri, ei;
```

```
    vector position;
    acid_label residue;
    int residue_num;
  } atom_info;
typedef struct rigid_unit_struct {
    unit_label type;
    connector head;
    int list_num;
    int n_bonds;
    bond_type **bond;
    int n_atoms;
    atom_info *atom;
  } rigid_unit;
typedef struct {
    atom_info *p;
    vector position;
  } atom_list;
typedef struct {
    char type1[NAME_LENGTH], type2[NAME_LENGTH],
         type3[NAME_LENGTH], type4[NAME_LENGTH];
    double v0[3], phi0[3];
  } torsion_data;
typedef struct torsion_list_struct {
    int num[4];
    torsion_data *p;
    int degen;
    struct torsion_list_struct *next;
  } torsion_list;
typedef struct {
    char type[NAME_LENGTH];
    double ri, ei;
  } lj_data;
typedef struct {
    char type1[NAME_LENGTH], type2[NAME_LENGTH];
    double a, b;
  } hbond_data;
typedef struct hbond_list_struct {
    int num[2];
    hbond_data *p;
```

```
                struct hbond_list_struct *next;
              } hbond_list;
typedef struct {
                rigid_unit *unit, *prev;
              } regrowth;

*************************************************************************
                GLOBAL VARIABLES - PEP_VAR.H
*************************************************************************

/* Global variables used in the program */
#if defined(MAIN)
#define EXT extern
#else
#define EXT
#endif
EXT torsion_data **torsion_data_list;
EXT lj_data **lj_data_list;
EXT hbond_data **hbond_data_list;
#undef EXT

*************************************************************************
                GLOBAL FUNCTIONS - PEPTIDE.H
*************************************************************************

/* Include files needed by peptide code */
#include <stdio.h>
#include <float.h>
#include <math.h>
#include <fcntl.h>
#include <stdio.h>
#include <memory.h>
#include <malloc.h>
#include <string.h>
#include <search.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <time.h>
```

```
#include <varargs.h>
/* global constants */
#define BETA 1.6886683 /* kB T at 298K */
#define MAX_BONDS 8
#define PI 3.1415927
#define EPS 1.0E-9
#define NAME_LENGTH 10
#define KMAX 100
#define MAX_ROOTS 100
#define DPHI .01
/* global macros */
#define INTERVAL(a,n1,n2) ((a) >= (n1) && (a) < (n2))
/* Include files relevant to this program */
#include "pep_type.h"
#include "pep_var.h"
/* random.c */
double ran(double dummy);
double ran2(double dummy);
/* peptide1.c */
void out_of_memory(void);
void get_sequence(string **sequence, int *n_peptides);
rigid_unit *read_peptide_data(string sequence, int *n_atoms_total,
                              int *max_atoms_per_unit);
rigid_unit *read_unit(string file, acid_label label, int
residue_num,
                      int *n_atoms_total, int *max_atoms_per_unit);
void couple_unit(rigid_unit *unit1, rigid_unit *unit2);
rigid_unit *modify_cystine_ends(rigid_unit *unit, int
n_amino_acids,
                                int *n_atoms_total);
void get_main_side(rigid_unit *unit, regrowth *main, regrowth
*side,
                   int *n_main, int *n_side);
void read_torsion_data(void);
void read_lj_data(void);
void read_hbond_data(void);
void write_car_file(int n_amino_acids, int n_atoms_total, atom_list
*atom,
                    string file);
```

```
string getline(string line, int len, FILE *fp);
void strip(string string);
void decomma(string string);
void capitalize(string s);
void amino_acid_code_3(acid_label label, string code_3);
void amino_acid_code_1(acid_label label, char code_1);
acid_label amino_acid_code(char code_1);
/* peptide2.c */
void initialize_connection_table(int **bond_table, int
n_atoms_total);
void make_connection_table(int **bond_table, int *table_num,
                           rigid_unit *unit, rigid_unit *start);
void add_connection(int **bond_table, int i1, int i2);
void print_connection_table(int **bond_table, int n_atoms_total);
void get_torsions(torsion_list **p, int **bond_table, int
*table_num,
                  atom_list *atom, rigid_unit *unit, rigid_unit
*start);
torsion_list *add_torsion(int **bond_table, atom_list *atom, int
i, int j,
                          int k, int l);
logical lookup_torsion_data(string type1, string type2, string
type3,
                            string type4, torsion_data **p);
void print_torsions(torsion_list *list, atom_list *atom);
double torsion(vector p1, vector p2, vector p3, vector p4);
void assign_lj_parameters(rigid_unit *unit, rigid_unit *start);
logical lookup_lj_data(string type, double *ri, double *ei);
logical lookup_lj_data(string type, double *ri, double *ei);
void get_hbonds(hbond_list **list, atom_list *atom, int n_atoms);
logical lookup_hbond_data(string type1, string type2, hbond_data
**p);
void print_hbonds(hbond_list *l, atom_list *atom);
void assign_atom_pointers(int *list_num, rigid_unit *unit,
rigid_unit *start,
                          atom_list *atom);
/* peptide3.c */
void old_unit(int *list_num, int n0, int n1, int n2, double
*logrosen,
```

```
                rigid_unit *unit, rigid_unit *start, torsion_list *t,
                hbond_list  *l,  atom_list  *atom,  vector  *twig[],
vector p0,
                vector b0);
void  do_unit(int  *list_num,  int  n0,  int  n1,  int  n2,  double
*logrosen,
              rigid_unit *unit, rigid_unit *start, torsion_list *t,
              hbond_list *l, atom_list *atom, vector *twig[], vector
p0,
              vector b0, double *e);
void do_backbone_f(int i, int n_main, int n_atoms_total,
                   double *logrosen,
                   regrowth *main, regrowth *side,
                   torsion_list *t, hbond_list *l,
                   atom_list *atom, vector *twig[],
                   double *e, logical new);
void do_backbone_f_rigid(int i, int n_main, int n_atoms_total,
                         double *logrosen,
                         regrowth *main, regrowth *side,
                         torsion_list *t, hbond_list *l,
                         atom_list *atom, atom_info *atom_tmp,
                         vector *twig[],
                         double *e, logical new);
void do_backbone_b(int i, int n_main, int n_atoms_total,
                   double *logrosen,
                   regrowth *main, regrowth *side,
                   torsion_list *t, hbond_list *l,
                   atom_list *atom, vector *twig[],
                   double *e, logical new);
void do_backbone_b_rigid(int i, int n_main, int n_atoms_total,
                         double *logrosen,
                         regrowth *main, regrowth *side,
                         torsion_list *t, hbond_list *l,
                         atom_list  *atom,  atom_info  *atom_tmp,
vector *twig[],
                         double *e, logical new);
void do_unit_sub(int *list_num, int n0, int n1, int n2, double
*logrosen,
                 rigid_unit *unit, torsion_list *t, hbond_list *l,
```

```
                atom_list *atom, vector *twig[], vector p1, vector
b1,
                vector  p0,  vector  b0,  double  *e,  vector
p[MAX_BONDS],
                vector b[MAX_BONDS], logical new);
void add_rigid_unit(rigid_unit *unit, vector *pos,
                    vector p1, vector b1, vector p0,
                    vector b0, vector point[MAX_BONDS],
                    vector bond[MAX_BONDS],
                    double cos_theta2, double sin_theta2);
vector align(vector p, vector r0, vector r1, vector n, double
cos_theta,
             double sin_theta, vector n2, double cos_theta2, double
sin_theta2);
/* peptide4.c */
double delta_energy(torsion_list *t, hbond_list *l, atom_list
*atom,
                    vector *twig, int n_atoms, int n0, int n1, int
n2,
                    int n_twig);
double energy(torsion_list *t, hbond_list *l, atom_list *atom,
              int n_atoms_total);
double d_nonbond_energy(torsion_list *t, atom_list *atom, vector
*twig,
                        int n_atoms, int n0, int n1, int n2, int
n_twig);
double  nonbond_energy(torsion_list  *t,  atom_list  *atom,  int
n_atoms_total);
double d_hbond_energy(hbond_list *l, atom_list *atom, vector *twig,
                      int n_atoms, int n0, int n1, int n2, int
n_twig);
double hbond_energy(hbond_list *l, atom_list *atom);
double d_torsion_energy(torsion_list *t, atom_list *atom, vector
*twig,
                      int n_atoms, int n0, int n1, int n2, int
n_twig);
double torsion_energy(torsion_list *t, atom_list *atom);
/* peptide5.c */
void do_mc(rigid_unit *unit, torsion_list *t, hbond_list *l,
```

```
            atom_list *atom, atom_list *atom2, atom_info *atom_tmp,
            vector *twig[], regrowth *main, regrowth *side,
            int n_amino_acids, int n_atoms_total, int n_main, int
n_side,
            logical cyclic);
void read_restart(atom_list *atom, int n_atoms_total);
void read_cycle(torsion_list *t, hbond_list *l,
                atom_list *atom, regrowth *main, regrowth *side,
                vector *twig[], int n_main, int n_side, int
n_atoms_total);
void regrow_main(torsion_list *t, hbond_list *l,
                 atom_list *atom, atom_list *atom2,
                 atom_info *atom_tmp, vector *twig[],
                 regrowth *main, regrowth *side,
                 int n_main, int n_atoms_total, double *e);
void regrow_side(torsion_list *t, hbond_list *l,
                 atom_list *atom, atom_list *atom2, vector *twig[],
                 regrowth *main, regrowth *side,
                 int n_side, int n_atoms_total, double *e);
/* peptide6.c */
void rotate_main(atom_list *atom, atom_list *atom2, vector *twig[],
                 regrowth *main, regrowth *side, torsion_list *t,
                 hbond_list *l, int n_main, int n_atoms_total,
double *e);
void get_rot_params(atom_list *atom, regrowth *main, int i0, int
n_main);
void get_rot_rosenbluth(atom_list *atom, atom_list *atom2,
                        vector *twig[], regrowth *main,
                        torsion_list *t, hbond_list *l, int i0, int
n_main,
                        int n_atoms_total, int *n, int *j, double
*logrosen,
                        double *e);
double jac(vector r[7]);
vector rotate_r1(atom_list *atom, regrowth *main, int i0, int
n_main);
void get_r(double phi1, double phi2, double phi3, double phi4,
double phi5);
void do_rotation(atom_list *atom, vector *twig, regrowth *main, int
```

```
i0,
                int n_main, int n_atoms_total);
void get_phi1(double phi[MAX_ROOTS][6], int *n);
void get_root(double x0, double x1, double *p1, double *p2, double
*p3,
              double *p4, double *p5, int n);
void F5init(vector q2, double *phi1);
void  F5(double  phi1,  double  phi2[4],  double  phi3[4],  double
phi4[4],
        double phi5[4], double f[4], logical valid[4]);
/* peptide7.c */
vector vector_rotate(vector a, vector n, double cos_theta, double
sin_theta);
vector get_main_b0(atom_list *atom, regrowth *main, int i);
vector get_main_p0(atom_list *atom, regrowth *main, int i);
vector get_side_b0(atom_list *atom, regrowth *side, int i);
vector get_side_p0(atom_list *atom, regrowth *side, int i);
void flory_rot_matrix(double theta, double phi, double m[3][3]);
vector flory_rot(double theta, double phi, vector a);
vector flory_rotinv(double theta, double phi, vector a);
void flory_lab(double m[3][3], vector r, vector l);
void flory_labinv(double m[3][3], vector r, vector l);
vector vector_cross(vector a, vector b);
vector vector_scale(vector v, double r);
void mxm(double m[3][3], double n[3][3]);
double det5(double m[5][5]);
double det(double m[3][3]);
vector mxb(double m[3][3], vector b);
vector bxm(double m[3][3], vector b);
double vector_dot(vector b1, vector b2);
double vector_length(vector v);
double vector_length2(vector v);

**************************************************************************
**************************************************************************
            DATA FILES DEFINING GEOMETRIC STRUCTURE
**************************************************************************
**************************************************************************
```

```
**********************************************************************
                DATA FILE FOR UNIT A - UNITA.DAT
**********************************************************************

! data file for rigid unit A--the NH2 terminus
1 !rigid unit in this structure
! ATOM INFORMATION
! rigid unit 0
3 !atoms in this rigid unit
N         0.039039567   -0.028048204    0.000005808 ALAn 1       NT
    N  -0.463
HN1      -0.294595420    0.946419656    0.000007165 ALAn 1        H
     H   0.126
HN2      -0.309849501   -0.509882152   -0.840834498 ALAn 1        H
     H   0.126
! BOND INFORMATION
! rigid unit 0
0 1 2 -1 -1 !ending of incoming bond--doesn't mean anything, but
must not be 1
0  0  .00000001!beginning of incoming bond -- just an overall
displacement
1 !bond out from this unit
-1 !don't know which unit this bond goes to
0 1 2 -1 -1 !beginning of outgoing backbone bond
1.498959541   -0.043336947   -0.000000042 !ending of outoing bond

**********************************************************************
                DATA FILE FOR UNIT B - UNITB.DAT
**********************************************************************

! data file for rigid unit B--the CH alpha carbon unit
1 !rigid unit in this structure
! ATOM INFORMATION
! rigid unit 0
2 !atoms in this rigid unit
CA        4.047343731    2.755753756   -0.000011837 ALA  2       CT
    C   0.035
HA        3.779272556    3.294512749   -0.928205431 ALA  2       HC
    H   0.032
```

```
! BOND INFORMATION
! rigid unit 0
0 1 -1 -1 -1!ending of incoming backbone bond
3.370934725    1.461895347    -0.000009674 !beginning of incoming
backbone bond
2 !bonds out from this unit
-1 !don't know which unit this bond goes to
0 1 -1 -1 -1 !beginning of outgoing side-chain bond
3.538550615      3.547572851      1.217100978 !ending of outgoin
side-chain bond
-1 !don't know which unit this bond goes to
0 1 -1 -1 -1!beginning of outgoing backbone bond
5.547336102      2.582198620     -0.000015057 !ending of outgoing
backbone bond

**********************************************************************
                DATA FILE FOR UNIT C - UNITC.DAT
**********************************************************************

! data file for rigid unit C--the OCNH amide bond unit
1 !rigid unit in this structure
! ATOM INFORMATION
! rigid unit 0
4 !atoms in this rigid unit
C         2.054825068     1.360626340     0.000001071 ALAn 1       C
     C    0.616
O         1.320880890     2.356072187     0.011419594 ALAn 1       O
     O   -0.504
N         3.370934725     1.461895347    -0.000009674 ALA  2       N
     N   -0.463
HN        3.917454243     0.530382395    -0.000003380 ALA  2       H
     H    0.252
! BOND INFORMATION
! rigid unit 0
0 1 2 -1 -1 !ending of incoming main-chain bond
1.498959541    -0.043336947    -0.000000042 !beginning of incoming
main-chain bond
1 !bond out from this unit
-1 !don't know which unit this bond goes to
```

```
2 0 3 -1 -1 !beginning of outgoing main-chain bond
4.047343731      2.755753756     -0.000011837 !ending of outging
main-chain bond

**********************************************************************
                DATA FILE FOR UNIT D - UNITD.DAT
**********************************************************************

! data file for rigid unit D--the HCO terminus
1 !rigid unit in this structure
! ATOM INFORMATION
! rigid unit 0
3 !atoms in this rigid unit
C         8.274295807     5.082911491    -0.000008575 ALAN 3       C
     C    0.616
HC        9.361082077     5.166533947    -0.000010758 ALAN 3       HC
    H    0.000
O         7.540351391     6.078356743     0.011415332 ALAN 3       O
     O   -0.504
! BOND INFORMATION
! rigid unit 0
0 1 2 -1 -1 !ending of incoming main-chain bond
7.718430996     3.678948641    -0.000013665 !beginning of incoming
main-chain bond
0 !bonds out from this unit

**********************************************************************
                DATA FILE FOR ALANINE - A.DAT
**********************************************************************

! The side-chain structure file for Alanine
1 !rigid unit in side-chain
! ATOM INFORMATION
! rigid unit 0
4 !atoms in this rigid unit
CB        3.178086281     3.790203094     1.217109203 ALA  2
CT       C  -0.098
HB1       3.502361059     4.845792770     1.274110079 ALA  2
HC       H    0.038
```

```
HB2       2.072028160     3.800241470     1.180677295 ALA  2
HC        H    0.038
HB3       3.465983868     3.309211969     2.172164917 ALA  2
HC        H    0.038
! BOND INFORMATION
! rigid unit 0
0 1 2 3 -1 !ending of incoming bond for unit 0 and nn
 3.783586502    3.069634676   -0.000003090 !beginning of bond for
unit 0
0 !bonds out from rigid unit 0

*****************************************************************
                DATA FILE FOR CYSTEINE - C.DAT
*****************************************************************

! The side-chain structure file for Cysteine
! Do not modify the atom order in this file
2 !rigid units in side-chain
! ATOM INFORMATION
! rigid unit 0
3 !atoms in this rigid unit
CB        3.185384274     3.813543320     1.210355163 CYSH 2
CT        C  -0.060
HB1       2.082855701     3.742515087     1.217666388 CYSH 2
HC        H    0.038
HB2       3.528102398     3.371057510     2.168041706 CYSH 2
HC        H    0.038
! rigid unit 1
4 !atoms in this rigid unit
SG        3.628824234     5.564641953     1.168115854 CYSH 2
SH        S    0.827
LG1       2.774378061     6.223292828     1.382826447 CYSH 2
LP        L  -0.481
LG2       4.018448353     5.879447937     0.188784361 CYSH 2
LP        L  -0.481
HG        4.543437004     5.521058083     2.133599997 CYSH 2
HS        H    0.135
! BOND INFORMATION
! rigid unit 0
```

```
0 1 2 -1 -1 !ending of incoming bond for unit 0 and nn
3.783586502   3.069634914   -0.000003354  !beginning of bond for
unit 0
1 !bonds out from rigid unit 0
1 !unit 0 is bonded to unit 1
0 1 2 -1 -1   ! beginning of outgoing bond and nn
 3.628824234   5.564641953   1.168115854  !ending of outgoing
bond for unit 0
! rigid unit 1
0 1 2 3 -1 !ending of incoming bond for unit 1 and nn
3.185384274   3.813543320   1.210355163 !beginning of bond for
unit 1
0 !bonds out from rigid unit 1

*****************************************************************
                DATA FILE FOR ASPARTATE - D.DAT
*****************************************************************

! The side-chain structure file for Aspartate
2 !rigid units in side-chain
! ATOM INFORMATION
! rigid unit 0
3 !atoms in this rigid unit
CB        3.195193052    3.859569550    1.198083878 ASP  2
CT      C  -0.398
HB1       2.099623203    3.734851122    1.256908774 ASP  2
HC      H   0.071
HB2       3.574837923    3.424842119    2.144523859 ASP  2
HC      H   0.071
! rigid unit 1
3 !atoms in this rigid unit
CG        3.488366127    5.366341114    1.240691185 ASP  2
C       C   0.714
OD1       3.752036572    5.965095997    2.273211718 ASP  2
O2      O  -0.721
OD2       3.445515871    5.949848175    0.005213364 ASP  2
O2      O  -0.721
! BOND INFORMATION
! rigid unit 0
```

```
0 1 2 -1 -1 !ending of incoming bond for unit 0 and nn
3.783586502   3.069634438   -0.000003352  !beginning of bond for
unit 0
1 !bonds out from rigid unit 0
1 !unit 0 is bonded to unit 1
0 1 2 -1 -1   ! beginning of outgoing bond and nn
 3.488366127    5.366341114    1.240691185  !ending of outgoing
bond for unit 0
! rigid unit 1
0 1 2 -1 -1 !ending of incoming bond for unit 1 and nn
3.195193052   3.859569550   1.198083878  !beginning of bond for
unit 1
0 !bonds out from rigid unit 1

*************************************************************
                DATA FILE FOR GLUTAMINE - E.DAT
*************************************************************

! The side-chain structure file for Glutamine
3 !rigid units in side-chain
! ATOM INFORMATION
! rigid unit 0
3 !atoms in this rigid unit
CB        3.210191727    3.806770086    1.242457986 GLU  2
CT      C  -0.184
HB1       3.453276873    4.884052753    1.160096049 GLU  2
HC      H   0.092
HB2       2.103818655    3.775332928    1.193925381 GLU  2
HC      H   0.092
! rigid unit 1
3 !atoms in this rigid unit
CG        3.670672178    3.303917646    2.650651217 GLU  2
CT      C  -0.398
HG1       3.495624304    2.214699984    2.732162237 GLU  2
HC      H   0.071
HG2       4.766538143    3.410970449    2.754028797 GLU  2
HC      H   0.071
! rigid unit 2
3 !atoms in this rigid unit
```

```
CD        3.044564962     3.944746017     3.891577959 GLU   2
C       C   0.714
OE1       3.318646908     3.594962835     5.031950951 GLU   2
O2      O  -0.721
OE2       2.157183647     4.937835217     3.607111931 GLU   2
O2      O  -0.721
! BOND INFORMATION
! rigid unit 0
0 1 2 -1 -1 !ending of incoming bond for unit 0 and nn
 3.783586502     3.069634438    -0.000003351  !beginning of bond
for unit 0
1 !bonds out from rigid unit 0
1 !unit 0 is bonded to unit 1
0 1 2 -1 -1    ! beginning of outgoing bond and nn
3.670672178     3.303917646     2.650651217   !ending of outgoing
bond for unit 0
! rigid unit 1
0 1 2 -1 -1 !ending of incoming bond for unit 1 and nn
3.210191727     3.806770086     1.242457986 !beginning of bond for
unit 1
1 !bonds out from rigid unit 1
2 !unit 1 is bonded to unit 2
0 1 2 -1 -1    ! beginning of outgoing bond and nn
 3.044564962     3.944746017     3.891577959   !ending of outgoing
bond for unit 1
! rigid unit 2
0 1 2 -1 -1 !ending of incoming bond for unit 1 and nn
3.670672178     3.303917646     2.650651217 !beginning of bond for
unit 2
0 !bonds out from rigid unit 2

************************************************************
              DATA FILE FOR PHENYLALANINE - F.DAT
************************************************************

! The side-chain structure file for Phenylalanine
2 !rigid units in side-chain
! ATOM INFORMATION
! rigid unit 0
```

```
3 !atoms in this rigid unit
CB        3.271046400     3.829343796     1.261018753 PHE  2
CT       C  -0.100
HB1       3.711064339     3.375446320     2.172759056 PHE  2
HC       H   0.108
HB2       3.680548668     4.858696938     1.261503935 PHE  2
HC       H   0.108
! rigid unit 1
11 !atoms in this rigid unit
CG        1.746863961     3.913921356     1.435816050 PHE  2
CA       C  -0.100
CD1       1.070973635     2.894981861     2.116770267 PHE  2
CA       C  -0.150
HD1       1.621361971     2.061387062     2.533305407 PHE  2
HC       H   0.150
CD2       1.019180536     4.963639259     0.869901121 PHE  2
CA       C  -0.150
HD2       1.528048277     5.750367641     0.331381440 PHE  2
HC       H   0.150
CE1      -0.315989435     2.915796280     2.214086056 PHE  2
CA       C  -0.150
HE1      -0.830357015     2.108316422     2.715482712 PHE  2
HC       H   0.150
CE2      -0.369023502     4.989082813     0.977358818 PHE  2
CA       C  -0.150
HE2      -0.928361893     5.798536777     0.531342983 PHE  2
HC       H   0.150
CZ       -1.036266327     3.964326382     1.646436572 PHE  2
CA       C  -0.150
HZ       -2.113304853     3.975853443     1.718335271 PHE  2
HC       H   0.150
! BOND INFORMATION
! rigid unit 0
0 1 2 -1 -1 !ending of incoming bond and nn
3.783586264    3.069634914   -0.000003353 !beginning of bond
1 !bonds out
1 !unit bonded to
0 1 2 -1 -1   ! beginning of outgoing bond and nn
1.746863961     3.913921356     1.435816050 !ending of outgoing
```

```
bond
! rigid unit 1
0 1 3 -1 -1 !ending of incoming bond and nn
3.271046400    3.829343796    1.261018753 !beginning of bond
0 !bonds out

*****************************************************************
                DATA FILE FOR GLYCINE - G.DAT
*****************************************************************

! The side-chain structure file for Glycine
1 !rigid unit in side-chain
! ATOM INFORMATION
! rigid unit 0
1 !atom in this rigid unit
HA2       2.054570675   -0.518772364   -0.887896836 GLYN 1
HC      H   0.032
! BOND INFORMATION
! rigid unit 0
0 -1 -1 -1 -1 !ending of incoming bond for unit 0 and nn
1.612465143   -0.031237146   -0.000000015 !beginning of incoming
bond for unit 0
0 !bonds out from rigid unit 0

*****************************************************************
                DATA FILE FOR HISTIDINE - H.DAT
*****************************************************************

! The side-chain structure file for Histidine
2 !rigid units in side-chain
! ATOM INFORMATION
! rigid unit 0
3 !atoms in this rigid unit
CB        3.239844084    3.731920242    1.277127385 HIS  2
CT      C  -0.098
HB1       2.644425392    3.025787830    1.893024564 HIS  2
HC      H   0.038
HB2       4.064783096    4.071127415    1.934927344 HIS  2
HC      H   0.038
```

```
! rigid unit 1
8 !atoms in this rigid unit
CG        2.370461226     4.918142319     0.978080690 HIS   2
CC      C   0.251
ND1       2.062596560     5.403582573    -0.290515751 HIS   2
NB      N  -0.502
CE1       1.272076607     6.440367222     0.045922592 HIS   2
CR      C   0.241
NE2       1.048720956     6.674089432     1.367565274 HIS   2
NA      N  -0.146
CD2       1.767608762     5.675839901     1.972463250 HIS   2
CW      C  -0.184
HE1       0.858503580     7.036557198    -0.757577479 HIS   2
HC      H   0.036
HE2       0.480951071     7.411210537     1.809884906 HIS   2
H       H   0.228
HD2       1.867301583     5.485908508     3.037219763 HIS   2
HC      H   0.114
! BOND INFORMATION
! rigid unit 0
0 1 2 -1 -1 !ending of incoming bond for unit 0 and nn
3.783586502    3.069634438   -0.000003353 !beginning of bond for
unit 0
1 !bonds out from rigid unit 0
1 !unit 0 is bonded to unit 1
0 1 2 -1 -1    ! beginning of outgoing bond and nn
2.370461226     4.918142319     0.978080690 !ending of outgoing
bond for unit 0
! rigid unit 1
0 1 4 -1 -1 !ending of incoming bond for unit 1 and nn
3.222899199    3.830397844    1.236912012 !beginning of bond for
unit 1
0 !bonds out from rigid unit 1

*****************************************************************
               DATA FILE FOR ISOLEUCINE - I.DAT
*****************************************************************

! The side-chain structure file for Isoleucine
```

```
4 !rigid units in side-chain
! ATOM INFORMATION
! rigid unit 0
2 !atoms in this rigid unit
CB        3.184130907     3.905461311     1.203313947 ILE  2
CT       C  -0.012
HB        3.579479933     3.448693275     2.135145664 ILE  2
HC       H   0.022
! rigid unit 1
4 !atoms in this rigid unit
CG2       3.632628202     5.399640560     1.184555411 ILE  2
CT       C  -0.085
HG21      3.256929159     5.962747097     2.057613134 ILE  2
HC       H   0.029
HG22      4.728721142     5.525658131     1.229067683 ILE  2
HC       H   0.029
HG23      3.277012348     5.929985046     0.281316549 ILE  2
HC       H   0.029
! rigid unit 2
3 !atoms in this rigid unit
CG1       1.625806093     3.868085861     1.310235620 ILE  2
CT       C  -0.049
HG11      1.169472456     4.395492077     0.450418025 ILE  2
HC       H   0.027
HG12      1.273633957     2.823534966     1.211708426 ILE  2
HC       H   0.027
! rigid unit 3
4 !atoms in this rigid unit
CD1       1.028863907     4.391342163     2.632859945 ILE  2
CT       C  -0.085
HD11     -0.068560459     4.262083530     2.654643297 ILE  2
HC       H   0.028
HD12      1.436750174     3.852109432     3.508637428 ILE  2
HC       H   0.028
HD13      1.222232699     5.468014240     2.787941933 ILE  2
HC       H   0.028
! BOND INFORMATION
! rigid unit 0
0 1 -1 -1 -1 !ending of incoming bond and nn
```

```
3.783586502    3.069634438   -0.000003350  !beginning of bond
2 !bonds out
1 !unit bonded to
0 1 -1 -1 -1   ! beginning of outgoing bond and nn
3.632628202    5.399640560    1.184555411   !ending of outgoing
bond
2 !unit bonded to
0 1 -1 -1 -1   ! beginning of outgoing bond and nn
 1.625806093    3.868085861    1.310235620   !ending of outgoing
bond
! rigid unit 1
0 1 2 3 -1 !ending of incoming bond and nn
 3.184130907    3.905461311    1.203313947 !beginning of incoming
bond
0! bonds out
! rigid unit 2
0 1 2 -1 -1 !ending of incoming bond and nn
 3.184130907    3.905461311    1.203313947 !beginning of incoming
bond
1 !bonds out
3 !unit bonded to
0 1 2 -1 -1    ! beginning of outgoing bond and nn
1.028863907    4.391342163    2.632859945   !ending of outgoing
bond
! rigid unit 3
0 1 2 3 -1 !ending of incoming bond and nn
1.625806093    3.868085861    1.310235620 !beginning of bond
0 !bonds out

**********************************************************************
                  DATA FILE FOR LYSINE - K.DAT
**********************************************************************

! The side-chain structure file for Lysine
5 !rigid units in side-chain
! ATOM INFORMATION
! rigid unit 0
3 !atoms in this rigid unit
CB        3.218223095    3.829745770    1.231236458 LYS  2
```

```
CT       C  -0.098
HB1       2.112416506     3.764609814     1.234413505 LYS   2
HC       H    0.038
HB2       3.536234617     3.317805290     2.163102627 LYS   2
HC       H    0.038
! rigid unit 1
3 !atoms in this rigid unit
CG        3.638167858     5.320005417     1.281187057 LYS   2
CT       C  -0.160
HG1       4.741127968     5.406830788     1.274424553 LYS   2
HC       H    0.116
HG2       3.295989990     5.833013058     0.360635072 LYS   2
HC       H    0.116
! rigid unit 2
3 !atoms in this rigid unit
CD        3.153400660     6.084614754     2.516160011 LYS   2
CT       C  -0.180
HD1       2.046517849     6.074027538     2.552636147 LYS   2
HC       H    0.122
HD2       3.501233101     5.571547031     3.435809374 LYS   2
HC       H    0.122
! rigid unit 3
3 !atoms in this rigid unit
CE        3.699187756     7.518018246     2.469964743 LYS   2
CT       C  -0.038
HE1       4.805956841     7.515174866     2.558616400 LYS   2
HC       H    0.098
HE2       3.475801945     8.000639915     1.495867610 LYS   2
HC       H    0.098
! rigid unit 4
4 !atoms in this rigid unit
NZ        3.098134756     8.306216240     3.560437918 LYS   2
N3       N  -0.138
HZ1       3.463554621     9.268757820     3.530759573 LYS   2
H3       H    0.294
HZ2       2.074491024     8.324481964     3.447653770 LYS   2
H3       H    0.294
HZ3       3.335658073     7.877095222     4.466163158 LYS   2
H3       H    0.294
```

```
! BOND INFORMATION
! rigid unit 0
0 1 2 -1 -1 !ending of incoming bond and nn
3.783586502    3.069634914   -0.000003353 !beginning of bond
1 !bonds out
1 !unit bonded to
0 1 2 -1 -1   ! beginning of outgoing bond and nn
3.638167858    5.320005417    1.281187057  !ending of outgoing
bond
! rigid unit 1
0 1 2 -1 -1 !ending of incoming bond and nn
 3.218223095    3.829745770    1.231236458!beginning of bond
1 !bonds out
2 !unit bonded to
0 1 2 -1 -1   ! beginning of outgoing bond and nn
3.153400660    6.084614754    2.516160011 !ending of outgoing
bond
! rigid unit 2
0 1 2 -1 -1 !ending of incoming bond and nn
3.638167858    5.320005417    1.281187057 !beginning of bond
1 !bonds out
3 !unit bonded to
0 1 2 -1 -1   ! beginning of outgoing bond and nn
3.699187756    7.518018246    2.469964743 !ending of outgoing
bond
! rigid unit 3
0 1 2 -1 -1 !ending of incoming bond and nn
3.153400660    6.084614754    2.516160011!beginning of bond
1 !bonds out
4 !unit bonded to
0 1 2 -1 -1   ! beginning of outgoing bond and nn
 3.098134756    8.306216240    3.560437918 !ending of outgoing
bond
! rigid unit 4
0 1 2 3 -1 !ending of incoming bond and nn
3.699187756    7.518018246    2.469964743!beginning of bond
0 !bonds out

**********************************************************************
```

DATA FILE FOR LEUCINE - L.DAT

****

```
! The side-chain structure file for Leucine
4 !rigid units in side-chain
! ATOM INFORMATION
! rigid unit 0
3 !atoms in this rigid unit
CB        3.217977524     3.860693455     1.213688374 LEU  2
CT      C  -0.061
HB1       3.617908239     3.413237095     2.146348953 LEU  2
HC      H   0.033
HB2       3.641148329     4.884153843     1.193638206 LEU  2
HC      H   0.033
! rigid unit 1
2 !atoms in this rigid unit
CG        1.676206470     3.974944353     1.357627273 LEU  2
CT      C  -0.010
HG        1.273801684     2.962582827     1.570222020 LEU  2
HC      H   0.031
! rigid unit 2
4 !atoms in this rigid unit
CD1       1.322771311     4.880306721     2.545703411 LEU  2
CT      C  -0.107
HD11      0.229164675     4.936426640     2.704123735 LEU  2
HC      H   0.034
HD12      1.758654118     4.507015228     3.491832256 LEU  2
HC      H   0.034
HD13      1.684926391     5.916738033     2.406197309 LEU  2
HC      H   0.034
! rigid unit 3
4 !atoms in this rigid unit
CD2       0.998154640     4.504262924     0.083184890 LEU  2
CT      C  -0.107
HD21     -0.093163513     4.622812748     0.214309067 LEU  2
HC      H   0.034
HD22      1.406615853     5.481475830    -0.234147355 LEU  2
HC      H   0.034
HD23      1.130140185     3.802904606    -0.761629283 LEU  2
```

```
HC      H   0.034
! BOND INFORMATION
! rigid unit 0
0 1 2 -1 -1 !ending of incoming bond and nn
3.783586502    3.069634438   -0.000003367!beginning of bond
1 !bonds out
1 !unit bonded to
0 1 2 -1 -1   ! beginning of outgoing bond and nn
1.676206470    3.974944353    1.357627273   !ending of outgoing
bond
! rigid unit 1
0 1 -1 -1 -1 !ending of incoming bond and nn
 3.184130907   3.905461311   1.203313947 !beginning of incoming
bond
2! bonds out
2 !unit bonded to
0 1 -1 -1 -1   ! beginning of outgoing bond and nn
1.322771311    4.880306721    2.545703411   !ending of outgoing
bond
3 !unit bonded to
0 1 -1 -1 -1   ! beginning of outgoing bond and nn
0.998154640    4.504262924    0.083184890  !ending of outgoing
bond
! rigid unit 2
0 1 2 3 -1 !ending of incoming bond and nn
1.676206470    3.974944353   1.357627273 !beginning of incoming
bond
0 !bonds out
! rigid unit 3
0 1 2 3 -1 !ending of incoming bond and nn
1.676206470    3.974944353    1.357627273 !beginning of bond
0 !bonds out

*****************************************************************
               DATA FILE FOR METHIONINE - M.DAT
*****************************************************************

! The side-chain structure file for Methionine
4 !rigid units in side-chain
```

```
! ATOM INFORMATION
! rigid unit 0
3 !atoms in this rigid unit
CB        3.219568014     3.840672970     1.225060582 MET  2
CT       C  -0.151
HB1       3.547865868     3.348565578     2.163037539 MET  2
HC       H    0.027
HB2       3.671003819     4.850576401     1.262409329 MET  2
HC       H    0.027
! rigid unit 1
3 !atoms in this rigid unit
CG        1.685955524     4.011272907     1.265707970 MET  2
CT       C  -0.054
HG1       1.291312337     4.382569790     0.302083224 MET  2
HC       H    0.0652
HG2       1.199923158     3.034499168     1.452733874 MET  2
HC       H    0.0652
! rigid unit 2
3 !atoms in this rigid unit
SD        1.234688163     5.162067413     2.574714422 MET  2
S        S    0.737
LD1       1.486726403     6.202064514     2.319993973 MET  2
LP       L  -0.381
LD2       1.747960329     4.937880516     3.521441460 MET  2
LP       L  -0.381
! rigid unit 3
4 !atoms in this rigid unit
CE       -0.532971203     4.837210655     2.617241383 MET  2
CT       C  -0.134
HE1      -0.987815082     4.991072178     1.622043610 MET  2
HC       H    0.0652
HE2      -1.033426285     5.510134220     3.335405111 MET  2
HC       H    0.0652
HE3      -0.725545764     3.794905424     2.929581165 MET  2
HC       H    0.0652
! BOND INFORMATION
! rigid unit 0
0 1 2 -1 -1 !ending of incoming bond and nn
3.783586502     3.069634438    -0.000003354 !beginning of bond
```

```
1 !bonds out
1 !unit bonded to
0 1 2 -1 -1    ! beginning of outgoing bond and nn
1.685955524     4.011272907     1.265707970 !ending of outgoing
bond
! rigid unit 1
0 1 2 -1 -1 !ending of incoming bond and nn
3.219568014    3.840672970    1.225060582 !beginning of bond
1 !bonds out
2 !unit bonded to
0 1 2 -1 -1    ! beginning of outgoing bond and nn
1.234688163     5.162067413     2.574714422 !ending of outgoing
bond
! rigid unit 2
0 1 2 -1 -1 !ending of incoming bond and nn
1.685955524    4.011272907    1.265707970 !beginning of bond
1 !bonds out
3 !unit bonded to
0 1 2 -1 -1    ! beginning of outgoing bond and nn
 -0.532971203    4.837210655    2.617241383 !ending of outgoing
bond
! rigid unit 3
0 1 2 3 -1 !ending of incoming bond and nn
1.234688163    5.162067413    2.574714422!beginning of bond
0 !bonds out

*****************************************************************
               DATA FILE FOR APSARAGINE - N.DAT
*****************************************************************

! The side-chain structure file for Asparagine
2 !rigid units in side-chain
! ATOM INFORMATION
! rigid unit 0
3 !atoms in this rigid unit
CB       3.222899199     3.830397844     1.236912012 ASN  2
CT       C  -0.086
HB1      3.611397266     3.364436865     2.163546562 ASN  2
HC       H   0.038
```

```
HB2       3.616078854     4.863478184     1.264652491 ASN   2
HC        H   0.038
! rigid unit 1
5 !atoms in this rigid unit
CG        1.698638678     3.892561436     1.381467938 ASN   2
C         C   0.675
OD1       1.085211635     3.155725241     2.139311790 ASN   2
O         O  -0.470
ND2       1.031797171     4.746669292     0.652490914 ASN   2
N         N  -0.867
HD21      0.019928589     4.602556705     0.711063743 ASN   2
H         H   0.344
HD22      1.562326550     5.282481670    -0.034363598 ASN   2
H         H   0.344
! BOND INFORMATION
! rigid unit 0
0 1 2 -1 -1 !ending of incoming bond for unit 0 and nn
3.783586502    3.069634438   -0.000003353  !beginning of bond for
unit 0
1 !bonds out from rigid unit 0
1 !unit 0 is bonded to unit 1
0 1 2 -1 -1    ! beginning of outgoing bond and nn
1.698638678     3.892561436     1.381467938  !ending of outgoing
bond for unit 0
! rigid unit 1
0 1 2 -1 -1 !ending of incoming bond for unit 1 and nn
3.222899199    3.830397844    1.236912012 !beginning of bond for
unit 1
0 !bonds out from rigid unit 1

**********************************************************************
                DATA FILE FOR GLUTAMINE - Q.DAT
**********************************************************************

! The side-chain structure file for Glutamine
3 !rigid units in side-chain
! ATOM INFORMATION
! rigid unit 0
3 !atoms in this rigid unit
```

```
CB        3.221223593     3.805351734     1.236027122 GLN  2
CT      C  -0.098
HB1       2.115758896     3.733683825     1.223282218 GLN  2
HC      H   0.038
HB2       3.538368225     3.258102417     2.148239136 GLN  2
HC      H   0.038
! rigid unit 1
3 !atoms in this rigid unit
CG        3.619170427     5.311230183     1.384292126 GLN  2
CT      C  -0.102
HG1       4.719832420     5.417502403     1.395145655 GLN  2
HC      H   0.057
HG2       3.298108339     5.879051685     0.491232127 GLN  2
HC      H   0.057
! rigid unit 2
5 !atoms in this rigid unit
CD        3.148421526     6.090956688     2.618209839 GLN  2
C       C   0.675
OE1       3.471138716     7.255728722     2.789397001 GLN  2
O       O  -0.470
NE2       2.408394814     5.500250816     3.521779537 GLN  2
N       N  -0.867
HE21      2.231919527     4.508390427     3.353902817 GLN  2
H       H   0.344
HE22      2.192787886     6.069860935     4.342392445 GLN  2
H       H   0.344
! BOND INFORMATION
! rigid unit 0
0 1 2 -1 -1 !ending of incoming bond for unit 0 and nn
3.783586502   3.069634438  -0.000003353  !beginning of bond for
unit 0
1 !bonds out from rigid unit 0
1 !unit 0 is bonded to unit 1
0 1 2 -1 -1   ! beginning of outgoing bond and nn
3.619170427     5.311230183     1.384292126 !ending of outgoing
bond for unit 0
! rigid unit 1
0 1 2 -1 -1 !ending of incoming bond for unit 1 and nn
3.221223593   3.805351734   1.236027122 !beginning of bond for
```

```
unit 1
1 !bonds out from rigid unit 0
2 !unit 1 is bonded to unit 2
0 1 2 -1 -1   ! beginning of outgoing bond and nn
3.148421526    6.090956688    2.618209839  !ending of outgoing
bond for unit 2
! rigid unit 2
0 1 2 -1 -1 !ending of incoming bond for unit 2 and nn
 3.619170427   5.311230183   1.384292126 !beginning of bond for
unit 2
0 !bonds out from rigid unit 2

*****************************************************************
                DATA FILE FOR ARGININE - R.DAT
*****************************************************************

! The side-chain structure file for Arginine
4 !rigid units in side-chain
! ATOM INFORMATION
! rigid unit 0
3 !atoms in this rigid unit
CB        3.207483053    3.819248199    1.232642174 ARG  2
CT      C  -0.080
HB1       2.121760130    3.616136551    1.319550753 ARG  2
HC      H   0.056
HB2       3.644849300    3.393733978    2.159598827 ARG  2
HC      H   0.056
! rigid unit 1
3 !atoms in this rigid unit
CG        3.412360668    5.357305527    1.216631651 ARG  2
CT      C  -0.103
HG1       4.487451553    5.614737511    1.132990837 ARG  2
HC      H   0.074
HG2       2.938670874    5.796108723    0.315252036 ARG  2
HC      H   0.074
! rigid unit 2
3 !atoms in this rigid unit
CD        2.850392818    6.038671017    2.471077681 ARG  2
CT      C  -0.228
```

```
HD1       1.769480824     5.816972256     2.580044270 ARG   2
HC        H    0.133
HD2       3.353989840     5.649005413     3.379585028 ARG   2
HC        H    0.133
! rigid unit 3
9 !atoms in this rigid unit
NE        3.069616079     7.502031326     2.345978022 ARG   2
N2        N   -0.324
HE        3.539865971     7.837357998     1.493146777 ARG   2
H3        H    0.269
CZ        2.710799694     8.413488388     3.240067959 ARG   2
CA        C    0.760
NH1       2.972572088     9.643490791     2.971310854 ARG   2
N2        N   -0.624
HH11      3.439955235     9.745957375     2.068439484 ARG   2
H3        H    0.361
HH12      2.697422743    10.348603249     3.651821136 ARG   2
H3        H    0.361
NH2       2.114365101     8.144207001     4.363539696 ARG   2
N2        N   -0.624
HH21      1.888047814     8.930854797     4.969158173 ARG   2
H3        H    0.361
HH22      1.947107434     7.146794796     4.499028206 ARG   2
H3        H    0.361
! BOND INFORMATION
! rigid unit 0
0 1 2 -1 -1 !ending of incoming bond for unit 0 and nn
3.783586502    3.069634914   -0.000003315 !beginning of bond for
unit 0
1 !bond out from rigid unit 0
1 !unit 0 is bonded to unit 1
0 1 2 -1 -1 ! beginning of outgoing bond and nn
3.412360668     5.357305527     1.216631651 !ending of outgoing
bond for unit 0
! rigid unit 1
0 1 2 -1 -1 !ending of incoming bond for unit 0 and nn
 3.207483053    3.819248199    1.232642174 !beginning of bond for
unit 1
1 !bond out from rigid unit 1
```

```
2 !unit 1 is bonded to unit 2
0 1 2 -1 -1 ! beginning of outgoing bond and nn
2.850392818     6.038671017     2.471077681 !ending of outgoing
bond
! rigid unit 2
0 1 2 -1 -1 !ending of incoming bond for unit 0 and nn
3.412360668    5.357305527    1.216631651 !beginning of bond for
unit 2
1 !bond out from rigid unit 2
3 !unit 2 is bonded to unit 3
0 1 2 -1 -1 ! beginning of outgoing bond and nn
3.069616079     7.502031326     2.345978022 !ending of outgoing
bond
! rigid unit 3
0 1 2 -1 -1 !ending of incoming bond for unit 0 and nn
2.850392818    6.038671017    2.471077681!beginning of bond for
unit 3
0 !bonds out from rigid unit 3

*****************************************************************
                DATA FILE FOR SERINE - S.DAT
*****************************************************************

! The side-chain structure file for Serine
2 !rigid units in side-chain
! ATOM INFORMATION
! rigid unit 0
3 !atoms in this rigid unit
CB        3.203660250     3.871555328     1.191825747 SER  2
CT      C   0.018
HB1       3.445731640     4.945727825     1.071671009 SER  2
HC      H   0.119
HB2       2.097403765     3.828571320     1.202566266 SER  2
HC      H   0.119
! rigid unit 1
2 !atoms in this rigid unit
OG        3.711599350     3.433972597     2.457015276 SER  2
OH      O  -0.550
HG        3.430009127     2.523327112     2.580434084 SER  2
```

```
HO      H   0.310
! BOND INFORMATION
! rigid unit 0
0 1 2 -1 -1 !ending of incoming bond for unit 0 and nn
 3.783586502    3.069634438   -0.000003353  !beginning of bond
for unit 0
1 !bonds out from rigid unit 0
1 !unit 0 is bonded to unit 1
0 1 2 -1 -1   ! beginning of outgoing bond and nn
3.711599350    3.433972597    2.457015276 !ending of outgoing
bond for unit 0
! rigid unit 1
0 1 -1 -1 -1 !ending of incoming bond for unit 1 and nn
 3.203660250    3.871555328    1.191825747  !beginning of bond
for unit 1
0 !bonds out from rigid unit 1

*****************************************************************
                DATA FILE FOR THREONINE - T.DA
*****************************************************************

! The side-chain structure file for Threonine
3 !rigid units in side-chain
! ATOM INFORMATION
! rigid unit 0
2 !atoms in this rigid unit
CB        3.220216751    3.864162445    1.226425409 THR  2
CT      C   0.170
HB        3.504307270    3.322291374    2.154003382 THR  2
HC      H   0.082
! rigid unit 1
2 !atoms in this rigid unit
OG1       1.802008867    3.940322876    1.161503792 THR  2
OH      O  -0.550
HG1       1.520381451    4.374082565    1.972538352 THR  2
HO      H   0.310
! rigid unit 2
4 !atoms in this rigid unit
CG2       3.680637360    5.331728935    1.361316323 THR  2
```

```
CT      C  -0.191
HG21     3.224400043     5.832503796     2.234619141 THR  2
HC      H   0.065
HG22     4.774106026     5.420624733     1.502453089 THR  2
HC      H   0.065
HG23     3.418393373     5.928008556     0.466874599 THR  2
HC      H   0.065
! BOND INFORMATION
! rigid unit 0
0 1 -1 -1 -1 !ending of incoming bond and nn
3.783586502    3.069634438    -0.000003353 !beginning of bond
2 !bonds out
1 !unit 0 is bonded
0 1 -1 -1 -1    ! beginning of outgoing bond and nn
1.802008867     3.940322876     1.161503792 !ending of outgoing
bond for unit 0
2 !unit 0 is bonded
0 1 -1 -1 -1    ! beginning of outgoing bond and nn
3.680637360     5.331728935     1.361316323  !ending of outgoing
bond for unit 0
! rigid unit 1
0 1 -1 -1 -1 !ending of incoming bond and nn
 3.220216751     3.864162445     1.226425409  !beginning of bond
for unit 1
0 !bonds out
! rigid unit 2
0 1 2 3 -1 !ending of incoming bond and nn
 3.220216751     3.864162445     1.226425409  !beginning of bond
for unit 1
0 !bonds out

**********************************************************************
                DATA FILE FOR VALINE - V.DAT
**********************************************************************

! The side-chain structure file for Valine
3 !rigid units in side-chain
! ATOM INFORMATION
! rigid unit 0
```

```
 2 !atoms in this rigid unit
 CB        3.211601496     3.852613449     1.247815728 VAL  2
 CT       C  -0.012
 HB        3.447319269     3.248452187     2.150032282 VAL  2
 HC       H    0.024
 ! rigid unit 1
 4 !atoms in this rigid unit
 CG1       1.676198244     4.045934200     1.217347741 VAL  2
 CT       C  -0.091
 HG11      1.351996183     4.697401524     0.384493083 VAL  2
 HC       H    0.031
 HG12      1.142809749     3.084587097     1.106773376 VAL  2
 HC       H    0.031
 HG13      1.300095797     4.498250008     2.155061245 VAL  2
 HC       H    0.031
 ! rigid unit 2
 4 !atoms in this rigid unit
 CG2       3.797980547     5.269292355     1.500991821 VAL  2
 CT       C  -0.091
 HG21      3.634918213     5.953960419     0.647068620 VAL  2
 HC       H    0.031
 HG22      3.359194279     5.751780510     2.395626068 VAL  2
 HC       H    0.031
 HG23      4.886912346     5.247161865     1.696415067 VAL  2
 HC       H    0.031
 ! BOND INFORMATION
 ! rigid unit 0
 0 1 -1 -1 -1 !ending of incoming bond and nn
 3.783586502     3.069634438    -0.000003354 !beginning of bond
 2 !bonds out
 1 !unit bonded to
 0 1 -1 -1 -1    ! beginning of outgoing bond and nn
  1.676198244     4.045934200     1.217347741!ending of outgoing
 bond
 2 !unit bonded to
 0 1 -1 -1 -1    ! beginning of outgoing bond and nn
  3.797980547     5.269292355     1.500991821!ending of outgoing
 bond
 ! rigid unit 1
```

```
0 1 2 3 -1 !ending of incoming bond and nn
3.211601496    3.852613449    1.247815728  !beginning of outgoing
bond
0 !bonds out
! rigid unit 2
0 1 2 3 -1 !ending of incoming bond and nn
3.211601496    3.852613449    1.247815728  !beginning of outgoing
bond
0 !bonds out
**************************************************************************
                DATA FILE FOR TRYPTOPHAN - W.DAT
**************************************************************************

! The side-chain structure file for Tryptophan
2 !rigid units in side-chain
! ATOM INFORMATION
! rigid unit 0
3 !atoms in this rigid unit
CB        3.247885227     3.809360981     1.256884575 TRP  2
CT      C  -0.098
HB1       3.555066347     3.270197153     2.175767183 TRP  2
HC      H   0.038
HB2       3.728011608     4.802421093     1.350249052 TRP  2
HC      H   0.038
! rigid unit 1
15 !atoms in this rigid unit
CG        1.731538415     4.025276661     1.276940465 TRP  2
C*      C  -0.135
CD1       0.792832434     3.205200195     1.936712861 TRP  2
CW      C   0.044
NE1      -0.527979255     3.628766537     1.692452073 TRP  2
NA      N  -0.352
CE2      -0.376119167     4.727549076     0.861387193 TRP  2
CN      C   0.154
CD2       0.994750261     4.975831032     0.602216363 TRP  2
CB      C   0.146
HD1       1.058894038     2.330861330     2.516448259 TRP  2
HC      H   0.093
HE1      -1.402328849     3.197247982     2.011827707 TRP  2
```

```
H       H   0.271
CE3      1.387488961     6.039774895    -0.250452638 TRP  2
CA      C  -0.173
HE3      2.430646658     6.226261139    -0.463923573 TRP  2
HC      H   0.086
CZ3      0.392907262     6.841813087    -0.810243368 TRP  2
CA      C  -0.066
HZ3      0.674497783     7.661212444    -1.455789328 TRP  2
HC      H   0.057
CH2     -0.963685811     6.602497578    -0.548699141 TRP  2
CA      C  -0.077
HH2     -1.710847259     7.243553162    -0.992942095 TRP  2
HC      H   0.074
CZ2     -1.364877820     5.549452305     0.277642310 TRP  2
CA      C  -0.168
HZ2     -2.410887718     5.363564491     0.470484644 TRP  2
HC      H   0.084
! BOND INFORMATION
! rigid unit 0
0 1 2 -1 -1 !ending of incoming bond and nn
 3.783586740    3.069634914   -0.000003497 !beginning of bond
1 !bonds out
1 !unit 0 is bonded
0 1 2 -1 -1   ! beginning of outgoing bond and nn
 1.731538415     4.025276661     1.276940465!ending of outgoing
bond for unit 0
! rigid unit 1
0 1 4 -1 -1 !ending of incoming bond and nn
 3.247885227     3.809360981     1.256884575  !beginning of bond
for unit 1
0 !bonds out

**********************************************************************
                DATA FILE FOR TYROSINE - Y.DAT
**********************************************************************

! The side-chain structure file for Tyrosine
3 !rigid units in side-chain
! ATOM INFORMATION
```

```
! rigid unit 0
3 !atoms in this rigid unit
CB        3.293353796     3.842515945     1.259159327 TYR  2
CT       C  -0.098
HB1       3.703839302     3.358918667     2.169649363 TYR  2
HC       H   0.038
HB2       3.749134064     4.852351665     1.277104497 TYR  2
HC       H   0.038
! rigid unit 1
10 !atoms in this rigid unit
CG        1.778211594     4.019127369     1.411828637 TYR  2
CA       C  -0.030
CD1       1.068759203     3.196300983     2.292453527 TYR  2
CA       C  -0.002
HD1       1.585003138     2.435774803     2.862824917 TYR  2
HC       H   0.064
CD2       1.095163584     4.989490032     0.672801077 TYR  2
CA       C  -0.002
HD2       1.629922271     5.630218983    -0.014210327 TYR  2
HC       H   0.064
CE1      -0.309100747     3.338460445     2.427857637 TYR  2
CA       C  -0.264
HE1      -0.845880806     2.691843510     3.105883360 TYR  2
HC       H   0.102
CZ       -0.983952701     4.304777145     1.686211467 TYR  2
C        C   0.462
CE2      -0.283983082     5.129064560     0.809688389 TYR  2
CA       C  -0.264
HE2      -0.814125061     5.873366833     0.234044328 TYR  2
HC       H   0.102
! rigid unit 1
2 !atoms in this rigid unit
OH       -2.337103367     4.443373203     1.815491915 TYR  2
OH       O  -0.528
HH       -2.648404837     3.798558235     2.453088284 TYR  2
HO       H   0.334
! BOND INFORMATION
! rigid unit 0
0 1 2 -1 -1 !ending of incoming bond and nn
```

```
3.783586264    3.069634914   -0.000003354 !beginning of bond
1 !bonds out
1 !unit bonded to
0 1 2 -1 -1   ! beginning of outgoing bond and nn
 1.778211594     4.019127369     1.411828637!ending of outgoing
bond for unit 0
! rigid unit 1
0 1 3 -1 -1 !ending of incoming bond and nn
 3.293353796     3.842515945     1.259159327  !beginning of bond
for unit 1
1 !bonds out
2 !unit bonded to
7 5 8 -1 -1   ! beginning of outgoing bond and nn
-2.337103367     4.443373203     1.815491915 !ending of outgoing
bond for unit 0
! rigid unit 2
0 1 -1 -1 -1 !ending of incoming bond and nn
-0.983952701     4.304777145     1.686211467  !beginning of bond
for unit 1
0 !bonds out

*********************************************************************
          DATA FILE FOR INITIAL PROTOTYPE - CX6C.CAR
*********************************************************************

!BIOSYM archive 3
PBC=OFF
!DATE Thu Mar  2 10:02:29 1995
SG        0.051616628     8.775964550     2.653307337 CYSn 1
S       S   0.824
LG1      -0.116704460     8.906803991     3.732450018 CYSn 1
LP      L  -0.405
LG2      -0.816371929     8.216369655     2.274560255 CYSn 1
LP      L  -0.405
CB        1.625257994     7.970290997     2.280061368 CYSn 1
CT      C  -0.098
HB1       1.743097230     7.117856362     2.972980432 CYSn 1
HC      H   0.050
HB2       2.457560406     8.667686711     2.506611212 CYSn 1
```

```
HC      H    0.050
CA        1.664891168     7.503978115     0.811322158 CYSn 1
CT      C    0.035
HA        2.715618613     7.453348875     0.469159517 CYSn 1
HC      H    0.032
N         0.954382540     8.512673633     0.003030230 CYSn 1
NT      N   -0.463
C         1.063568189     6.132700222     0.616111991 CYSn 1
C       C    0.616
O         0.248707622     5.654726837     1.414398016 CYSn 1
O       O   -0.504
N         1.449902196     5.479885680    -0.464156147 GLY  2
N       N   -0.463
HN        2.157106102     5.992384244    -1.099457509 GLY  2
H       H    0.252
CA        0.868490592     4.154014497    -0.652902307 GLY  2
CT      C    0.035
HA1       1.550908149     3.403064022    -0.212395307 GLY  2
HC      H    0.032
HA2      -0.097660558     4.132736815    -0.116611463 GLY  2
HC      H    0.032
C         0.730531165     3.827591429    -2.120728786 GLY  2
C       C    0.616
O         1.559375145     4.206208097    -2.957020570 GLY  2
O       O   -0.504
N        -0.320742949     3.103195380    -2.456098946 GLY  3
N       N   -0.463
HN       -0.976177839     2.817016114    -1.646836012 GLY  3
H       H    0.252
CA       -0.454134161     2.787581074    -3.875321662 GLY  3
CT      C    0.035
HA1      -0.907422830     1.783240810    -3.972773051 GLY  3
HC      H    0.032
HA2      -1.127648566     3.540414569    -4.323795441 GLY  3
HC      H    0.032
C         0.896974016     2.736484179    -4.547627543 GLY  3
C       C    0.616
O         1.315189212     1.712629073    -5.101282348 GLY  3
O       O   -0.504
```

```
N      1.599575272    3.853622667   -4.520184621 GLY  4
N      N  -0.463
HN     1.137216234    4.691535216   -4.019658253 GLY  4
H      H   0.252
CA     2.905944550    3.804217731   -5.170228610 GLY  4
CT     C   0.035
HA1    3.056204584    2.789614618   -5.584558431 GLY  4
HC     H   0.032
HA2    2.897891721    4.540755026   -5.994216851 GLY  4
HC     H   0.032
C      4.014980067    4.050747291   -4.175561433 GLY  4
C      C   0.616
O      4.978871195    4.780583329   -4.436272241 GLY  4
O      O  -0.504
N      3.887759074    3.450944950   -3.006608050 GLY  5
N      N  -0.463
HN     3.003276191    2.844372268   -2.879487738 GLY  5
H      H   0.252
CA     4.960071382    3.689311240   -2.044877031 GLY  5
CT     C   0.035
HA1    5.709592998    2.881830301   -2.144167698 GLY  5
HC     H   0.032
HA2    5.427393718    4.658369322   -2.297948016 GLY  5
HC     H   0.032
C      4.437174470    3.643619035   -0.629041435 GLY  5
C      C   0.616
O      3.798322352    2.676595378   -0.197242766 GLY  5
O      O  -0.504
N      4.713663113    4.691871185    0.124033264 GLY  6
N      N  -0.463
HN     5.286002166    5.476492875   -0.348403798 GLY  6
H      H   0.252
CA     4.208080753    4.647691975    1.492986659 GLY  6
CT     C   0.035
HA1    3.303800182    4.010943092    1.515218779 GLY  6
HC     H   0.032
HA2    4.993057374    4.194323221    2.125265975 GLY  6
HC     H   0.032
C      3.799265981    6.023038258    1.963510280 GLY  6
```

```
C       C   0.616
O         4.006824522    7.036283245    1.285298717 GLY  6
O       O  -0.504
N         3.195690211    6.077750863    3.136158080 GLY  7
N       N  -0.463
HN        3.055107813    5.133307510    3.640799839 GLY  7
H       H   0.252
CA        2.800412417    7.407555656    3.591101372 GLY  7
CT      C   0.035
HA1       1.946687677    7.303619509    4.286815466 GLY  7
HC      H   0.032
HA2       3.660862081    7.847316876    4.127520148 GLY  7
HC      H   0.032
C         2.334578164    8.258959996    2.434291753 GLY  7
C       C   0.616
O         2.337411236    9.494643783    2.487154063 GLY  7
O       O  -0.504
N         1.936206121    7.605756209    1.358640986 CYSN 8
N       N  -0.463
HN        1.983632457    6.528240768    1.414418956 CYSN 8
H       H   0.252
CA        1.485796919    8.428968216    0.240136508 CYSN 8
CT      C   0.035
HA        0.399931102    8.271042216    0.100059529 CYSN 8
HC      H   0.032
C         2.167493478    8.018162291   -1.043072620 CYSN 8
C       C   0.616
CB        1.746659419    9.902481747    0.610166221 CYSN 8
CT      C  -0.098
HB1       2.709270705   10.016688002    1.140264476 CYSN 8
HC      H   0.050
HB2       1.816139488   10.541353385   -0.293951287 CYSN 8
HC      H   0.050
SG        0.440719361   10.532225816    1.688457720 CYSN 8
S       S   0.824
LG1      -0.404239097   10.957145937    1.126774557 CYSN 8
LP      L  -0.405
LG2       0.793091788   11.329491558    2.359427872 CYSN 8
LP      L  -0.405
```

```
end
end

**********************************************************************
**********************************************************************
                        END OF LISTING
**********************************************************************
**********************************************************************

**********************************************************************
**********************************************************************
             DATA FILE WEINER FORCES - AMBER.FRC
**********************************************************************
**********************************************************************

!BIOSYM forcefield          2
#version amber.frc  1.0  19-Oct-90
#version amber.frc  1.1   8-Aug-92
#define amber
> This is the new format version of the amber forcefield
!Ver  Ref           Function        Label
!---- ---   ---------------------------------     ------
 1.0   1     atom_types                    amber
 1.0   1     equivalence                   amber
 1.0   1     hbond_definition              amber
 1.0   1     quadratic_bond                amber
 1.0   1     quadratic_angle               amber
 1.0   1     torsion_3                     amber
 1.0   1     out_of_plane                  amber
 1.0   1     nonbond(12-6)                 amber
 1.0   1     hydrogen_bond(10-12)          amber
#atom_types    amber
> Atom type definitions for any variant of amber
> Masses from CRC 1973/74 pages B-250.
!Ver  Ref  Type    Mass       Element  Comment
!---- ---  ----  ----------  -------
 1.0   1    C     12.000000     C       Kollman's Field: Masses
from CRC 1973/74 pages B-250.
 1.0   1    C*    12.000000     C
```

1.0	1	C2	12.000000	C
1.0	3	C3	15.000000	C
1.0	1	CA	12.000000	C
1.0	1	CB	12.000000	C
1.0	1	CC	12.000000	C
1.0	3	CD	13.000000	C
1.0	3	CE	13.000000	C
1.0	3	CF	13.000000	C
1.0	3	CG	13.000000	C
1.0	3	CH	13.000000	C
1.0	3	CI	13.000000	C
1.0	3	CJ	13.000000	C
1.0	1	CK	12.000000	C
1.0	1	CM	12.000000	C
1.0	1	CN	12.000000	C
1.0	3	CP	13.000000	C
1.0	1	CQ	12.000000	C
1.0	1	CR	12.000000	C
1.0	1	CT	12.000000	C
1.0	1	CV	12.000000	C
1.0	1	CW	12.000000	C
1.0	1	H	1.007825	H
1.0	1	H2	1.007825	H
1.0	1	H3	1.007825	H
1.0	1	HC	1.007825	H
1.0	1	HO	1.007825	H
1.0	1	HS	1.007825	H
1.0	3	LP	3.000000	H
1.0	1	N	14.003070	N
1.0	1	N*	14.003070	N
1.0	1	N2	14.003070	N
1.0	1	N3	14.003070	N
1.0	1	NA	14.003070	N
1.0	1	NB	14.003070	N
1.0	1	NC	14.003070	N
1.0	1	NP	14.003070	N
1.0	1	NT	14.003070	N
1.0	1	O	15.994910	O
1.0	1	O2	15.994910	O

```
1.0   1   OH    15.994910   O
1.0   1   OS    15.994910   O
1.0   1   P     30.993760   P
1.0   1   S     31.972070   S
1.0   1   SH    31.972070   S
1.0   3   C0    40.080000   Ca
1.0   3   HW    1.008000    H
1.0   3   IM    35.450000   Cl
1.0   3   CU    63.550000   Cu
1.0   3   I     22.990000   I
1.0   3   MG    24.305000   Mg
1.0   3   OW    16.000000   O
1.0   3   QC    132.90000   Cs
1.0   3   QK    39.100000   K
1.0   3   QL    6.940000    Li
1.0   3   QN    22.990000   Na
1.0   3   QR    85.470000   Rb
1.1   4   CS    12.000000   C     carbohydrate sp3 carbon
1.1   4   AC    12.000000   C     carbohydrate alpha-anomeric
carbon
1.1   4   BC    12.000000   C     carbohydrate beta-anomeric
carbon
1.1   4   HT    1.007825    H     carbohydrate sp3 hydro
1.1   4   AH    1.007825    H     carbohydrate alpha-anomeric
hydrogen
1.1   4   BH    1.007825    H     carbohydrate beta-anomeric
hydrogen
1.1   4   HY    1.007825    H     carbohydrate hydroxyl
hydrogen
1.1   4   OT    15.994910   O     carbohydrate hydroxyl
oxygen
1.1   4   OA    15.994910   O     carbohydrate alpha-anomeric
oxygen
1.1   4   OB    15.994910   O     carbohydrate beta-anomeric
oxygen
1.1   4   OE    15.994910   O     carbohydrate ring oxygen
1.0   1   h$    1.007825    H     Hydrogen atom for aTOMATIC
PARAMETER assignment
1.0   1   c$    12.000000   C     Carbon atom for automatic
```

parameter assignment
1.0 1 n$ 14.003070 N Nitrogen atom for automatic
parameter assignment
1.0 1 o$ 15.994910 O Oxygen atom for automatic
parameter assignment
1.0 1 s$ 31.972070 S Sulfur atom for automatic
parameter assignment
1.0 1 p$ 30.993760 P Phosphorous atom for
automatic parameter assignment
#equivalence amber
> Equivalence table for any variant of amber

!		Equivalences					
!Ver	Ref	Type	NonB	Bond	Angle	Torsion	OOP
1.0	1	C	C	C	C	C	C
1.0	1	C*	C*	C*	C*	C*	C*
1.0	1	C2	C2	C2	C2	C2	C2
1.0	1	C3	C3	C3	C3	C3	C3
1.0	1	CA	CA	CA	CA	CA	CA
1.0	1	CB	CB	CB	CB	CB	CB
1.0	1	CC	CC	CC	CC	CC	CC
1.0	1	CD	CD	CD	CD	CD	CD
1.0	1	CE	CE	CE	CE	CE	CE
1.0	1	CF	CF	CF	CF	CF	CF
1.0	1	CG	CG	CG	CG	CG	CG
1.0	1	CH	CH	CH	CH	CH	CH
1.0	1	CI	CI	CI	CI	CI	CI
1.0	1	CJ	CJ	CJ	CJ	CJ	CJ
1.0	1	CK	CK	CK	CK	CK	CK
1.0	1	CM	CM	CM	CM	CM	CM
1.0	1	CN	CN	CN	CN	CN	CN
1.0	1	CP	CP	CP	CP	CP	CP
1.0	1	CQ	CQ	CQ	CQ	CQ	CQ
1.0	1	CR	CR	CR	CR	CR	CR
1.0	1	CT	CT	CT	CT	CT	CT
1.0	1	CV	CV	CV	CV	CV	CV
1.0	1	CW	CW	CW	CW	CW	CW
1.0	1	H	H	H	H	H	H

1.0	1	H2	H2	H2	H2	H2	H2
1.0	1	H3	H3	H3	H3	H3	H3
1.0	1	HC	HC	HC	HC	HC	HC
1.0	1	HO	HO	HO	HO	HO	HO
1.0	1	HS	HS	HS	HS	HS	HS
1.0	1	LP	LP	LP	LP	LP	LP
1.0	1	N	N	N	N	N	N
1.0	1	N*	N*	N*	N*	N*	N*
1.0	1	N2	N2	N2	N2	N2	N2
1.0	1	N3	N3	N3	N3	N3	N3
1.0	1	NA	NA	NA	NA	NA	NA
1.0	1	NB	NB	NB	NB	NB	NB
1.0	1	NC	NC	NC	NC	NC	NC
1.0	1	NP	NP	NP	NP	NP	NP
1.0	1	NT	NT	NT	NT	NT	NT
1.0	1	O	O	O	O	O	O
1.0	1	O2	O2	O2	O2	O2	O2
1.0	1	OH	OH	OH	OH	OH	OH
1.0	1	OS	OS	OS	OS	OS	OS
1.0	1	P	P	P	P	P	P
1.0	1	S	S	S	S	S	S
1.0	1	SH	SH	SH	SH	SH	SH
1.0	3	I	I	I	I	I	I
1.0	3	CU	CU	CU	CU	CU	CU
1.0	3	IM	IM	IM	IM	IM	IM
1.0	3	C0	C0	C0	C0	C0	C0
1.0	3	HW	HW	HW	HW	HW	HW
1.0	3	MG	MG	MG	MG	MG	MG
1.0	3	OW	OW	OW	OW	OW	OW
1.0	3	QC	QC	QC	QC	QC	QC
1.0	3	QK	QK	QK	QK	QK	QK
1.0	3	QL	QL	QL	QL	QL	QL
1.0	3	QN	QN	QN	QN	QN	QN
1.0	3	QR	QR	QR	QR	QR	QR
1.1	4	CS	CS	CS	CS	CS	CS
1.1	4	AC	AC	AC	AC	AC	AC
1.1	4	BC	BC	BC	BC	BC	BC
1.1	4	HT	HT	HT	HT	HT	HT
1.1	4	AH	AH	AH	AH	AH	AH

```
 1.1   4    BH    BH      BH      BH      BH      BH
 1.1   4    HY    HY      HY      HY      HY      HY
 1.1   4    OT    OT      OT      OT      OT      OT
 1.1   4    OA    OA      OA      OA      OA      OA
 1.1   4    OB    OB      OB      OB      OB      OB
 1.1   4    OE    OE      OE      OE      OE      OE
 1.0   1    h$    h$      h$      h$      h$      h$
 1.0   1    c$    c$      c$      c$      c$      c$
 1.0   1    n$    n$      n$      n$      n$      n$
 1.0   1    o$    o$      o$      o$      o$      o$
 1.0   1    s$    s$      s$      s$      s$      s$
 1.0   1    p$    p$      p$      p$      p$      p$
#hbond_definition   amber
 1.0   1   distance      2.5000
 1.0   1   angle        90.0000
 1.0   1   donors        H   HO  H2  H3  HS
 1.0   1   acceptors     NB  NC  O2  O   OH  S   SH
#quadratic_bond     amber
> E = K2 * (R - R0)^2
!Ver  Ref     I     J          R0          K2
!---- ---    ----  ----     -------    --------
 1.0   3     OW    HW        0.9572    553.0000
 1.0   3     HW    HW        1.5136    553.0000
 1.0   3     CH    N3        1.471     367.0000
 1.0   3     C3    SH        1.810     222.0000
 1.0   1     C     C2        1.5220    317.0000
 1.0   1     C     C3        1.5220    317.0000
 1.0   1     C     CA        1.4000    469.0000
 1.0   1     C     CB        1.4190    447.0000
 1.0   1     C     CD        1.4000    469.0000
 1.0   1     C     CH        1.5220    317.0000
 1.0   1     C     CJ        1.4440    410.0000
 1.0   1     C     CM        1.4440    410.0000
 1.0   3     C     CT        1.5220    317.0000
 1.0   1     C     N         1.3350    490.0000
 1.0   1     C     N*        1.3830    424.0000
 1.0   1     C     NA        1.3880    418.0000
 1.0   1     C     NC        1.3580    457.0000
 1.0   1     C     O         1.2290    570.0000
```

1.0	1	C	O2	1.2500	656.0000
1.0	1	C	OH	1.3640	450.0000
1.0	1	C*	C2	1.4950	317.0000
1.0	1	C*	CB	1.4590	388.0000
1.0	1	C*	CG	1.3520	546.0000
1.0	1	C*	CT	1.4950	317.0000
1.0	1	C*	CW	1.3520	546.0000
1.0	1	C*	HC	1.0800	340.0000
1.0	1	C2	C2	1.5260	260.0000
1.0	1	C2	C3	1.5260	260.0000
1.0	1	C2	CA	1.5100	317.0000
1.0	1	C2	CC	1.5040	317.0000
1.0	1	C2	CH	1.5260	260.0000
1.0	1	C2	N	1.4490	337.0000
1.0	1	C2	N2	1.4630	337.0000
1.0	1	C2	N3	1.4710	367.0000
1.0	1	C2	NT	1.4710	367.0000
1.0	1	C2	OH	1.4250	386.0000
1.0	1	C2	OS	1.4250	320.0000
1.0	1	C2	S	1.8100	222.0000
1.0	1	C2	SH	1.8100	222.0000
1.0	1	C3	CH	1.5260	260.0000
1.0	1	C3	CM	1.5100	317.0000
1.0	1	C3	N	1.4490	337.0000
1.0	1	C3	N*	1.4750	337.0000
1.0	1	C3	N2	1.4630	337.0000
1.0	1	C3	N3	1.4710	367.0000
1.0	1	C3	OH	1.4250	386.0000
1.0	1	C3	OS	1.4250	320.0000
1.0	1	C3	S	1.8100	222.0000
1.0	1	CA	CA	1.4000	469.0000
1.0	1	CA	CB	1.4040	469.0000
1.0	1	CA	CD	1.4000	469.0000
1.0	1	CA	CJ	1.4330	427.0000
1.0	1	CA	CM	1.4330	427.0000
1.0	1	CA	CN	1.4000	469.0000
1.0	1	CA	CT	1.5100	317.0000
1.0	1	CA	HC	1.0800	340.0000
1.0	1	CA	N2	1.3400	481.0000

1.0	1	CA	NA	1.3810	427.0000
1.0	1	CA	NC	1.3390	483.0000
1.0	1	CB	CB	1.3700	520.0000
1.0	1	CB	CD	1.4000	469.0000
1.0	1	CB	CN	1.4190	447.0000
1.0	1	CB	N*	1.3740	436.0000
1.0	1	CB	NB	1.3910	414.0000
1.0	1	CB	NC	1.3540	461.0000
1.0	1	CC	CF	1.3750	512.0000
1.0	1	CC	CG	1.3710	518.0000
1.0	1	CC	CT	1.5040	317.0000
1.0	1	CC	CV	1.3750	512.0000
1.0	1	CC	CW	1.3710	518.0000
1.0	1	CC	NA	1.3850	422.0000
1.0	1	CC	NB	1.3940	410.0000
1.0	1	CD	CD	1.4000	469.0000
1.0	1	CD	CN	1.4000	469.0000
1.0	1	CE	N*	1.3710	440.0000
1.0	1	CE	NB	1.3040	529.0000
1.0	1	CF	NB	1.3940	410.0000
1.0	1	CG	NA	1.3810	427.0000
1.0	1	CH	CH	1.5260	260.0000
1.0	1	CH	N	1.4490	337.0000
1.0	1	CH	N*	1.4750	337.0000
1.0	1	CH	NT	1.4710	367.0000
1.0	1	CH	OH	1.4250	386.0000
1.0	1	CH	OS	1.4250	320.0000
1.0	1	CI	NC	1.3240	502.0000
1.0	1	CJ	CJ	1.3500	549.0000
1.0	1	CJ	CM	1.3500	549.0000
1.0	1	CJ	N*	1.3650	448.0000
1.0	1	CK	HC	1.0800	340.0000
1.0	1	CK	N*	1.3710	440.0000
1.0	1	CK	NB	1.3040	529.0000
1.0	1	CM	CM	1.3500	549.0000
1.0	1	CM	CT	1.5100	317.0000
1.0	1	CM	HC	1.0800	340.0000
1.0	1	CM	N*	1.3650	448.0000
1.0	1	CN	NA	1.3800	428.0000

1.0	1	CP	NA	1.3430	477.0000
1.0	1	CP	NB	1.3350	488.0000
1.0	1	CQ	HC	1.0800	340.0000
1.0	1	CQ	NC	1.3240	502.0000
1.0	1	CR	HC	1.0800	340.0000
1.0	1	CR	NA	1.3430	477.0000
1.0	1	CR	NB	1.3350	488.0000
1.0	1	CT	CT	1.5260	310.0000
1.0	1	CT	HC	1.0900	331.0000
1.0	3	CT	N	1.4490	337.0000
1.0	1	CT	N*	1.4750	337.0000
1.0	1	CT	N2	1.4630	337.0000
1.0	1	CT	N3	1.4710	367.0000
1.0	1	CT	OH	1.4100	320.0000
1.0	1	CT	OS	1.4100	320.0000
1.0	1	CT	S	1.8100	222.0000
1.0	1	CT	SH	1.8100	222.0000
1.0	1	CV	HC	1.0800	340.0000
1.0	1	CV	NB	1.3940	410.0000
1.0	1	CW	HC	1.0800	340.0000
1.0	1	CW	NA	1.3810	427.0000
1.0	1	H	N	1.0100	434.0000
1.0	1	H	N2	1.0100	434.0000
1.0	1	H	NA	1.0100	434.0000
1.0	1	H	N*	1.0100	434.0000
1.0	1	H2	N	1.0100	434.0000
1.0	1	H2	N2	1.0100	434.0000
1.0	1	H2	NT	1.0100	434.0000
1.0	1	H3	N2	1.0100	434.0000
1.0	1	H3	N3	1.0100	434.0000
1.0	1	HO	OH	0.9600	553.0000
1.0	1	HO	OS	0.9600	553.0000
1.0	1	HS	SH	1.3360	274.0000
1.0	3	LP	S	0.6790	150.0000
1.0	3	LP	SH	0.6790	150.0000
1.0	1	O2	P	1.4800	525.0000
1.0	1	OH	P	1.6100	230.0000
1.0	1	OS	P	1.6100	230.0000
1.0	1	S	S	2.0380	166.0000

1.1	4	OH	HO	0.9600	553.0000
1.1	4	OT	HY	0.9720	460.5000
1.1	4	OA	HY	0.9720	460.5000
1.1	4	OB	HY	0.9720	460.5000
1.1	4	CS	HT	1.0990	337.3000
1.1	4	AC	AH	1.0990	337.3000
1.1	4	BC	BH	1.0990	337.3000
1.1	4	AC	HT	1.0990	337.3000
1.1	4	BC	HT	1.0990	337.3000
1.1	4	AC	OA	1.4110	334.3000
1.1	4	BC	OB	1.3900	334.3000
1.1	4	CS	OA	1.4400	334.3000
1.1	4	CS	OB	1.4400	334.3000
1.1	4	CS	CS	1.5230	214.8000
1.1	4	CS	CT	1.5230	214.8000
1.1	4	AC	CS	1.5230	214.8000
1.1	4	BC	CS	1.5230	214.8000
1.1	4	CS	OT	1.4110	334.3000
1.1	4	CS	OE	1.4270	296.7000
1.1	4	AC	OE	1.4270	296.7000
1.1	4	BC	OE	1.4270	296.7000
1.1	4	CS	N	1.4490	355.0000
1.1	4	H	N	1.0100	434.0000
1.1	4	C	N	1.3350	490.0000
1.1	4	C	O	1.2290	570.0000
1.1	4	C	CS	1.5220	335.0000
1.0	1	C$1	C$1	1.5260	260.0000
1.0	1	C$2	C$2	1.4000	469.0000
1.0	1	C$3	C$3	1.3700	520.0000
1.0	1	C$5	C$5	1.2040	590.0000
1.0	1	C$1	O$1	1.4250	386.0000
1.0	1	C$2	O$2	1.2500	280.0000
1.0	1	C$3	O$3	1.2300	300.0000
1.0	1	C$1	N$1	1.4490	337.0000
1.0	1	C$2	N$2	1.3810	427.0000
1.0	1	C$5	N$5	1.1580	649.0000
1.0	1	C$1	S$1	1.8100	222.0000
1.0	1	C$1	H$1	1.0900	331.0000

```
 1.0   1     O$1   O$1       1.4800     590.0000
 1.0   1     O$3   O$3       1.2080     590.0000
 1.0   1     O$1   N$1       1.2400     300.0000
 1.0   1     O$2   N$2       1.1900     450.0000
 1.0   1     O$3   N$3       1.1860     590.0000
 1.0   1     O$1   H$1       0.9600     553.0000
 1.0   1     N$1   N$1       1.1300     300.0000
 1.0   1     N$1   H$1       1.0100     434.0000
 1.0   1     S$1   S$1       2.0380     166.0000
 1.0   1     S$1   H$1       1.3360     274.0000
 1.0   1     O$1   P$1       1.6100     230.0000
 1.0   1     O$2   P$2       1.4800     525.0000
 1.0   1     P$1   H$1       1.5000     200.0000
#quadratic_angle    amber
> E = K2 * (Theta - Theta0)^2
!Ver  Ref     I     J     K        Theta0          K2
!---- ---    ----  ----  ----     --------      -------
 1.0   3     HW    OW    HW       104.5200     100.0000
 1.0   3     O     C     O        126.0000      80.0000
 1.0   3     C     CH    N3       109.7000      80.0000
 1.0   3     CH    CH    N3       109.7000      80.0000
 1.0   3     C     CT    N3       112.0000      80.0000
 1.0   3     CH    N3    H3       109.5000      35.0000
 1.0   3     CT    N3    CT       113.0000      50.0000
 1.0   3     P     OS    P        120.5000     100.0000
 1.0   1     C     C2    C2       112.4000      63.0000
 1.0   1     C     C2    CH       112.4000      63.0000
 1.0   1     C     C2    N        110.3000      80.0000
 1.0   1     C     C2    NT       111.2000      80.0000
 1.0   1     C     CA    CA       120.0000      85.0000
 1.0   1     C     CA    HC       120.0000      35.0000
 1.0   1     C     CB    CB       119.2000      85.0000
 1.0   1     C     CB    NB       130.0000      70.0000
 1.0   1     C     CD    CD       120.0000      85.0000
 1.0   1     C     CH    C2       111.1000      63.0000
 1.0   1     C     CH    C3       111.1000      63.0000
 1.0   1     C     CH    CH       111.1000      63.0000
 1.0   1     C     CH    N        110.1000      63.0000
 1.0   1     C     CH    NT       109.7000      80.0000
```

1.0	1	C	CJ	CJ	120.7000	85.0000
1.0	1	C	CM	C3	119.7000	85.0000
1.0	1	C	CM	CJ	120.7000	85.0000
1.0	1	C	CM	CM	120.7000	85.0000
1.0	1	C	CM	CT	119.7000	70.0000
1.0	1	C	CM	HC	119.7000	35.0000
1.0	1	C	CT	CT	111.1000	63.0000
1.0	1	C	CT	HC	109.5000	35.0000
1.0	1	C	CT	N	110.1000	63.0000
1.0	1	C	N	C2	121.9000	50.0000
1.0	1	C	N	C3	121.9000	50.0000
1.0	1	C	N	CH	121.9000	50.0000
1.0	1	C	N	CT	121.9000	50.0000
1.0	1	C	N	H	119.8000	35.0000
1.0	1	C	N	H2	120.0000	35.0000
1.0	1	C	N*	CH	117.6000	70.0000
1.0	1	C	N*	CJ	121.6000	70.0000
1.0	1	C	N*	CM	121.6000	70.0000
1.0	1	C	N*	CT	117.6000	70.0000
1.0	1	C	N*	H	119.2000	35.0000
1.0	1	C	NA	C	126.4000	70.0000
1.0	1	C	NA	CA	125.2000	70.0000
1.0	1	C	NA	H	116.8000	35.0000
1.0	1	C	NC	CA	120.5000	70.0000
1.0	1	C	OH	HO	113.0000	35.0000
1.0	1	C*	C2	CH	115.6000	63.0000
1.0	1	C*	CB	CA	134.9000	85.0000
1.0	1	C*	CB	CD	134.9000	85.0000
1.0	1	C*	CB	CN	108.8000	85.0000
1.0	1	C*	CG	NA	108.7000	70.0000
1.0	1	C*	CT	HC	109.5000	35.0000
1.0	1	C*	CW	HC	120.0000	35.0000
1.0	1	C*	CW	NA	108.7000	70.0000
1.0	1	C2	C	N	116.6000	70.0000
1.0	1	C2	C	O	120.4000	80.0000
1.0	1	C2	C	O2	117.0000	70.0000
1.0	1	C2	C*	CB	128.6000	70.0000
1.0	1	C2	C*	CG	125.0000	70.0000
1.0	1	C2	C*	CW	125.0000	70.0000

1.0	1	C2	C2	C2	112.4000	63.0000
1.0	1	C2	C2	CH	112.4000	63.0000
1.0	1	C2	C2	N	111.2000	80.0000
1.0	1	C2	C2	N2	111.2000	80.0000
1.0	1	C2	C2	N3	111.2000	80.0000
1.0	1	C2	C2	NT	111.2000	80.0000
1.0	1	C2	C2	OS	109.5000	80.0000
1.0	1	C2	C2	S	114.7000	50.0000
1.0	1	C2	CA	CA	120.0000	70.0000
1.0	1	C2	CA	CD	120.0000	70.0000
1.0	1	C2	CC	CF	131.9000	70.0000
1.0	1	C2	CC	CG	129.0000	70.0000
1.0	1	C2	CC	CV	131.9000	70.0000
1.0	1	C2	CC	CW	129.0000	70.0000
1.0	1	C2	CC	NA	122.2000	70.0000
1.0	1	C2	CC	NB	121.0000	70.0000
1.0	1	C2	CH	C3	111.5000	63.0000
1.0	1	C2	CH	CH	111.5000	63.0000
1.0	1	C2	CH	N	109.7000	80.0000
1.0	1	C2	CH	N*	109.5000	80.0000
1.0	1	C2	CH	NT	109.7000	80.0000
1.0	1	C2	CH	OH	109.5000	80.0000
1.0	1	C2	CH	OS	109.5000	80.0000
1.0	1	C2	N	CH	118.0000	50.0000
1.0	1	C2	N	H	118.4000	38.0000
1.0	1	C2	N2	CA	123.2000	50.0000
1.0	1	C2	N2	H2	118.4000	35.0000
1.0	1	C2	N2	H3	118.4000	35.0000
1.0	1	C2	N3	H3	109.5000	35.0000
1.0	1	C2	NT	H2	109.5000	35.0000
1.0	1	C2	OH	HO	108.5000	55.0000
1.0	1	C2	OS	C2	111.8000	100.0000
1.0	1	C2	OS	C3	111.8000	100.0000
1.0	1	C2	OS	HO	108.5000	55.0000
1.0	1	C2	OS	P	120.5000	100.0000
1.0	1	C2	S	C3	98.9000	62.0000
1.0	3	C2	S	LP	96.7000	150.0000
1.0	1	C2	S	S	103.7000	68.0000
1.0	1	C2	SH	HS	96.0000	44.0000

1.0	3	C2	SH	LP	96.7000	150.0000
1.0	1	C3	C	N	116.6000	70.0000
1.0	1	C3	C	O	120.4000	80.0000
1.0	1	C3	C	O2	117.0000	70.0000
1.0	1	C3	C2	CH	112.4000	63.0000
1.0	1	C3	C2	OS	109.5000	80.0000
1.0	1	C3	CH	C3	111.5000	63.0000
1.0	1	C3	CH	CH	111.5000	63.0000
1.0	1	C3	CH	N	109.5000	80.0000
1.0	1	C3	CH	NT	109.7000	80.0000
1.0	1	C3	CH	OH	109.5000	80.0000
1.0	1	C3	CM	CJ	119.7000	85.0000
1.0	1	C3	N	H	118.4000	38.0000
1.0	1	C3	N*	CB	125.8000	70.0000
1.0	1	C3	N*	CE	128.8000	70.0000
1.0	1	C3	N*	CK	128.8000	70.0000
1.0	1	C3	N2	CA	123.2000	50.0000
1.0	1	C3	N2	H2	118.4000	35.0000
1.0	1	C3	N3	H3	109.5000	35.0000
1.0	1	C3	OH	HO	108.5000	55.0000
1.0	1	C3	OS	P	120.5000	100.0000
1.0	3	C3	S	LP	96.7000	150.0000
1.0	1	C3	S	S	103.7000	68.0000
1.0	1	C3	SH	HS	96.0000	44.0000
1.0	3	C3	SH	LP	96.7000	150.0000
1.0	1	CA	C	CA	120.0000	85.0000
1.0	1	CA	C	OH	120.0000	70.0000
1.0	1	CT	C	OH	117.0000	70.0000
1.0	3	CT	C	O2	117.0000	70.0000
1.0	1	CA	C2	CH	114.0000	63.0000
1.0	1	CA	CA	CA	120.0000	85.0000
1.0	1	CA	CA	CB	120.0000	85.0000
1.0	1	CA	CA	CN	120.0000	85.0000
1.0	1	CA	CA	CT	120.0000	70.0000
1.0	1	CA	CA	HC	120.0000	35.0000
1.0	1	CA	CB	CB	117.3000	85.0000
1.0	1	CA	CB	CN	116.2000	85.0000
1.0	1	CA	CB	NB	132.4000	70.0000
1.0	1	CA	CD	CD	120.0000	85.0000

1.0	1	CA	CJ	CJ	117.0000	85.0000
1.0	1	CA	CM	CM	117.0000	85.0000
1.0	1	CA	CM	HC	123.3000	35.0000
1.0	1	CA	CN	CB	122.7000	85.0000
1.0	1	CA	CN	NA	132.8000	70.0000
1.0	1	CA	CT	CT	114.0000	63.0000
1.0	1	CA	CT	HC	109.5000	35.0000
1.0	1	CA	N2	CT	123.2000	50.0000
1.0	1	CA	N2	H	120.0000	35.0000
1.0	1	CA	N2	H2	120.0000	35.0000
1.0	1	CA	N2	H3	120.0000	35.0000
1.0	1	CA	NA	H	118.0000	35.0000
1.0	1	CA	NC	CB	112.2000	70.0000
1.0	1	CA	NC	CI	118.6000	70.0000
1.0	1	CA	NC	CQ	118.6000	70.0000
1.0	1	CB	C	NA	111.3000	70.0000
1.0	1	CB	C	O	128.8000	80.0000
1.0	1	CB	C*	CG	106.4000	85.0000
1.0	1	CB	C*	CT	128.6000	70.0000
1.0	1	CB	C*	CW	106.4000	85.0000
1.0	1	CB	C*	HC	126.8000	35.0000
1.0	1	CB	CA	HC	120.0000	35.0000
1.0	1	CB	CA	N2	123.5000	70.0000
1.0	1	CB	CA	NC	117.3000	70.0000
1.0	1	CB	CB	N*	106.2000	70.0000
1.0	1	CB	CB	NB	110.4000	70.0000
1.0	1	CB	CB	NC	127.7000	70.0000
1.0	1	CB	CD	CD	120.0000	85.0000
1.0	1	CB	CN	CD	122.7000	85.0000
1.0	1	CB	CN	NA	104.4000	70.0000
1.0	1	CB	N*	CE	105.4000	70.0000
1.0	1	CB	N*	CH	125.8000	70.0000
1.0	1	CB	N*	CK	105.4000	70.0000
1.0	1	CB	N*	CT	125.8000	70.0000
1.0	1	CB	N*	H	127.3000	35.0000
1.0	1	CB	NB	CE	103.8000	70.0000
1.0	1	CB	NB	CK	103.8000	70.0000
1.0	1	CB	NC	CI	111.0000	70.0000
1.0	1	CB	NC	CQ	111.0000	70.0000

1.0	1	CC	C2	CH	113.1000	63.0000
1.0	1	CC	CF	NB	109.9000	70.0000
1.0	1	CC	CG	NA	105.9000	70.0000
1.0	1	CC	CT	CT	113.1000	63.0000
1.0	1	CC	CT	HC	109.5000	35.0000
1.0	1	CC	CV	HC	120.0000	35.0000
1.0	1	CC	CV	NB	109.9000	70.0000
1.0	1	CC	CW	HC	120.0000	35.0000
1.0	1	CC	CW	NA	105.9000	70.0000
1.0	1	CC	NA	CP	107.3000	70.0000
1.0	1	CC	NA	CR	107.3000	70.0000
1.0	1	CC	NA	H	126.3000	35.0000
1.0	1	CC	NB	CP	105.3000	70.0000
1.0	1	CC	NB	CR	105.3000	70.0000
1.0	1	CD	C	CD	120.0000	85.0000
1.0	1	CD	C	OH	120.0000	70.0000
1.0	1	CD	CA	CD	120.0000	85.0000
1.0	1	CD	CB	CN	116.2000	85.0000
1.0	1	CD	CD	CD	120.0000	85.0000
1.0	1	CD	CD	CN	120.0000	85.0000
1.0	1	CD	CN	NA	132.8000	70.0000
1.0	1	CE	N*	CH	128.8000	70.0000
1.0	1	CE	N*	CT	128.8000	70.0000
1.0	1	CE	N*	H	127.3000	35.0000
1.0	1	CF	CC	NA	105.9000	70.0000
1.0	1	CF	NB	CP	105.3000	70.0000
1.0	1	CF	NB	CR	105.3000	70.0000
1.0	1	CG	CC	NA	108.7000	70.0000
1.0	1	CG	CC	NB	109.9000	70.0000
1.0	1	CG	NA	CN	111.6000	70.0000
1.0	1	CG	NA	CP	107.3000	70.0000
1.0	1	CG	NA	CR	107.3000	70.0000
1.0	1	CG	NA	H	126.3000	35.0000
1.0	1	CH	C	N	116.6000	70.0000
1.0	1	CH	C	O	120.4000	80.0000
1.0	1	CH	C	O2	117.0000	65.0000
1.0	1	CH	C	OH	115.0000	70.0000
1.0	1	CH	C2	CH	112.4000	63.0000
1.0	1	CH	C2	OH	109.5000	80.0000

1.0	1	CH	C2	OS	109.5000	80.0000
1.0	1	CH	C2	S	114.7000	50.0000
1.0	1	CH	C2	SH	108.6000	50.0000
1.0	1	CH	CH	CH	111.5000	63.0000
1.0	1	CH	CH	N	109.7000	80.0000
1.0	1	CH	CH	N*	109.5000	80.0000
1.0	1	CH	CH	NT	109.7000	80.0000
1.0	1	CH	CH	OH	109.5000	80.0000
1.0	1	CH	CH	OS	109.5000	80.0000
1.0	1	CH	N	H	118.4000	38.0000
1.0	1	CH	N*	CJ	121.2000	70.0000
1.0	1	CH	N*	CK	128.8000	70.0000
1.0	1	CH	NT	H2	109.5000	35.0000
1.0	1	CH	OH	HO	108.5000	55.0000
1.0	1	CH	OS	CH	111.8000	100.0000
1.0	1	CH	OS	HO	108.5000	55.0000
1.0	1	CH	OS	P	120.5000	100.0000
1.0	1	CJ	C	NA	114.1000	70.0000
1.0	1	CJ	C	O	125.3000	80.0000
1.0	1	CJ	CA	N2	120.1000	70.0000
1.0	1	CJ	CA	NC	121.5000	70.0000
1.0	1	CJ	CJ	N*	121.2000	70.0000
1.0	1	CJ	CM	CT	119.7000	85.0000
1.0	1	CJ	N*	CT	121.2000	70.0000
1.0	1	CJ	N*	H	119.2000	35.0000
1.0	1	CK	N*	CT	128.8000	70.0000
1.0	1	CM	C	NA	114.1000	70.0000
1.0	1	CM	C	O	125.3000	80.0000
1.0	1	CM	CA	N2	120.1000	70.0000
1.0	1	CM	CA	NC	121.5000	70.0000
1.0	1	CM	CJ	N*	121.2000	70.0000
1.0	1	CM	CM	CT	119.7000	70.0000
1.0	1	CM	CM	HC	119.7000	35.0000
1.0	1	CM	CM	N*	121.2000	70.0000
1.0	1	CM	CT	HC	109.5000	35.0000
1.0	1	CM	N*	CT	121.2000	70.0000
1.0	1	CM	N*	H	119.2000	35.0000
1.0	1	CN	CA	HC	120.0000	35.0000
1.0	1	CN	NA	CW	111.6000	70.0000

1.0	1	CN	NA	H	123.1000	35.0000
1.0	1	CP	NA	H	126.3000	35.0000
1.0	1	CR	NA	CW	107.3000	70.0000
1.0	1	CR	NA	H	126.3000	35.0000
1.0	1	CR	NB	CV	105.3000	70.0000
1.0	1	CT	C	N	116.6000	70.0000
1.0	1	CT	C	O	120.4000	80.0000
1.0	1	CT	C*	CW	125.0000	70.0000
1.0	1	CT	CC	CV	131.9000	70.0000
1.0	1	CT	CC	CW	129.0000	70.0000
1.0	1	CT	CC	NA	122.2000	70.0000
1.0	1	CT	CC	NB	121.0000	70.0000
1.0	1	CT	CT	CT	109.5000	40.0000
1.0	1	CT	CT	C*	115.6000	63.0000
1.0	1	CT	CT	HC	109.5000	35.0000
1.0	1	CT	CT	N	109.7000	80.0000
1.0	1	CT	CT	N*	109.5000	50.0000
1.0	1	CT	CT	N2	111.2000	80.0000
1.0	1	CT	CT	N3	111.2000	80.0000
1.0	1	CT	CT	OH	109.5000	50.0000
1.0	1	CT	CT	OS	109.5000	50.0000
1.0	1	CT	CT	S	114.7000	50.0000
1.0	1	CT	CT	SH	108.6000	50.0000
1.0	1	CT	N	CT	118.0000	50.0000
1.0	1	CT	N	H	118.4000	38.0000
1.0	1	CT	N2	H3	118.4000	35.0000
1.0	1	CT	N3	H3	109.5000	35.0000
1.0	1	CT	OH	HO	108.5000	55.0000
1.0	1	CT	OS	CT	109.5000	60.0000
1.0	1	CT	OS	P	120.5000	100.0000
1.0	1	CT	S	CT	98.9000	62.0000
1.0	3	CT	S	LP	96.7000	150.0000
1.0	1	CT	S	S	103.7000	68.0000
1.0	1	CT	SH	HS	96.0000	44.0000
1.0	3	CT	SH	LP	96.7000	150.0000
1.0	1	CV	CC	NA	105.9000	70.0000
1.0	1	CW	C*	HC	126.8000	35.0000
1.0	1	CW	CC	NA	108.7000	70.0000
1.0	1	CW	CC	NB	109.9000	70.0000

1.0	1	CW	NA	H	125.3000	35.0000
1.0	1	H	N	H	120.0000	35.0000
1.0	1	H2	N2	H2	120.0000	35.0000
1.0	1	H2	NT	H2	109.5000	35.0000
1.0	1	H3	N	H3	120.0000	35.0000
1.0	1	H3	N2	H3	120.0000	35.0000
1.0	1	H3	N3	H3	109.5000	35.0000
1.0	1	HC	CK	N*	123.0000	35.0000
1.0	1	HC	CK	NB	123.0000	35.0000
1.0	1	HC	CM	N*	119.1000	35.0000
1.0	1	HC	CQ	NC	115.4000	35.0000
1.0	1	HC	CR	NA	120.0000	35.0000
1.0	1	HC	CR	NB	120.0000	35.0000
1.0	1	HC	CT	HC	109.5000	35.5000
1.0	1	HC	CT	N	109.5000	38.0000
1.0	1	HC	CT	N*	109.5000	35.0000
1.0	1	HC	CT	N2	109.5000	35.0000
1.0	1	HC	CT	N3	109.5000	35.0000
1.0	1	HC	CT	OH	109.5000	35.0000
1.0	1	HC	CT	OS	109.5000	35.0000
1.0	1	HC	CT	S	109.5000	35.0000
1.0	1	HC	CT	SH	109.5000	35.0000
1.0	1	HC	CV	NB	120.0000	35.0000
1.0	1	HC	CW	NA	120.0000	35.0000
1.0	1	HO	OH	HO	104.5000	47.0000
1.0	1	HO	OH	P	108.5000	45.0000
1.0	1	HS	SH	HS	92.1000	35.0000
1.0	3	HS	SH	LP	96.7000	150.0000
1.0	3	LP	S	LP	160.0000	150.0000
1.0	3	LP	S	S	96.7000	150.0000
1.0	3	LP	SH	LP	160.0000	150.0000
1.0	1	N	C	O	122.9000	80.0000
1.0	1	N*	C	NA	115.4000	70.0000
1.0	1	N*	C	NC	118.6000	70.0000
1.0	1	N*	C	O	120.9000	80.0000
1.0	1	N*	CB	NC	126.2000	70.0000
1.0	1	N*	CE	NB	113.9000	70.0000
1.0	1	N*	CH	OS	109.5000	80.0000
1.0	1	N*	CK	NB	113.9000	70.0000

1.0	1	N*	CT	OS	109.5000	50.0000
1.0	1	N2	CA	N2	120.0000	70.0000
1.0	1	N2	CA	NA	116.0000	70.0000
1.0	1	N2	CA	NC	119.3000	70.0000
1.0	1	NA	C	O	120.6000	80.0000
1.0	1	NA	CA	NC	123.3000	70.0000
1.0	1	NA	CP	NA	110.7000	70.0000
1.0	1	NA	CP	NB	111.6000	70.0000
1.0	1	NA	CR	NA	110.7000	70.0000
1.0	1	NA	CR	NB	111.6000	70.0000
1.0	1	NC	C	O	122.5000	80.0000
1.0	1	NC	CI	NC	129.1000	70.0000
1.0	1	NC	CQ	NC	129.1000	70.0000
1.0	1	O	C	O2	126.0000	80.0000
1.0	1	O	C	OH	126.0000	80.0000
1.0	1	O2	C	O2	126.0000	80.0000
1.0	1	O2	P	O2	119.9000	140.0000
1.0	1	O2	P	OH	108.2000	45.0000
1.0	1	O2	P	OS	108.2000	100.0000
1.0	1	OH	P	OS	102.6000	45.0000
1.0	1	OS	P	OS	102.6000	45.0000
1.1	4	HO	OH	HO	104.5000	47.0000
1.1	4	CS	OT	HY	109.3500	53.6000
1.1	4	AC	OA	HY	109.3500	53.6000
1.1	4	BC	OB	HY	109.3500	53.6000
1.1	4	CS	OT	CS	117.0000	60.0000
1.1	4	AC	OA	CS	115.0000	62.0000
1.1	4	BC	OB	CS	116.4000	62.0000
1.1	4	CS	OE	AC	113.8000	90.7000
1.1	4	CS	OE	BC	111.9000	90.7000
1.1	4	HT	CS	HT	107.8500	33.6000
1.1	4	AH	AC	HT	107.8500	33.6000
1.1	4	BH	BC	HT	107.8500	33.6000
1.1	4	HT	CS	CS	108.7200	43.0000
1.1	4	HC	CT	CS	108.7200	43.0000
1.1	4	HT	CS	CT	108.7200	43.0000
1.1	4	AH	AC	CS	108.7200	43.0000
1.1	4	BH	BC	CS	108.7200	43.0000
1.1	4	HT	CS	AC	108.7200	43.0000

1.1	4	HT	CS	BC	108.7200	43.0000
1.1	4	HT	CS	OT	109.8900	45.9000
1.1	4	AH	AC	OA	109.8900	45.9000
1.1	4	BH	BC	OB	109.8900	45.9000
1.1	4	HT	AC	OA	109.8900	45.9000
1.1	4	HT	BC	OB	109.8900	45.9000
1.1	4	HT	CS	OA	109.8900	45.9000
1.1	4	HT	CS	OB	109.8900	45.9000
1.1	4	HT	CS	OE	107.2400	45.2000
1.1	4	HT	CS	C	109.5000	35.0000
1.1	4	AH	AC	OE	107.2400	45.2000
1.1	4	BH	BC	OE	107.2400	45.2000
1.1	4	HT	AC	OE	107.2400	45.2000
1.1	4	HT	BC	OE	107.2400	45.2000
1.1	4	CS	CS	CS	110.7000	38.0000
1.1	4	CS	CS	CT	110.7000	38.0000
1.1	4	CS	CS	AC	110.7000	38.0000
1.1	4	CS	CS	BC	110.7000	38.0000
1.1	4	CS	CS	OT	110.1000	75.7000
1.1	4	CS	CT	OH	110.1000	75.7000
1.1	4	CS	CS	OA	110.1000	75.7000
1.1	4	CS	CS	OB	110.1000	75.7000
1.1	4	CS	C	O	120.4000	80.0000
1.1	4	AC	CS	OT	110.1000	75.7000
1.1	4	BC	CS	OT	110.1000	75.7000
1.1	4	BC	CS	OB	110.1000	75.7000
1.1	4	BC	CS	OA	110.1000	75.7000
1.1	4	AC	CS	OB	110.1000	75.7000
1.1	4	AC	CS	OA	110.1000	75.7000
1.1	4	CS	AC	OA	110.1000	75.7000
1.1	4	CS	BC	OB	110.1000	75.7000
1.1	4	CS	CS	OE	109.4000	81.0000
1.1	4	CT	CS	OE	109.4000	81.0000
1.1	4	CS	AC	OE	109.4000	81.0000
1.1	4	CS	BC	OE	109.4000	81.0000
1.1	4	CS	OE	CS	113.8000	90.7000
1.1	4	OE	CS	OT	111.5500	92.6000
1.1	4	OE	AC	OA	111.5500	92.6000
1.1	4	OE	BC	OB	107.4000	92.6000

```
  1.1    4      BC      CS      N          109.7000      80.0000
  1.1    4      CS      CS      N          109.7000      80.0000
  1.1    4      HT      CS      N          109.5000      38.0000
  1.1    4      CS      N       H          118.4000      38.0000
  1.1    4      CS      N       C          121.9000      50.0000
  1.1    4      C       N       H          119.8000      35.0000
  1.1    4      N       C       O          122.9000      80.0000
  1.1    4      N       C       CS         116.6000      70.0000
  1.0    1      $$      C$4     $$         109.5000      63.0000
  1.0    1      $$      C$3     $$         120.0000      85.0000
  1.0    1      $$      C$2     $$         180.0000     200.0000
  1.0    1      $$      O$2     $$         109.5000     100.0000
  1.0    1      $$      N$4     $$         109.5000      60.0000
  1.0    1      $$      N$3     $$         114.0000      60.0000
  1.0    1      $$      N$2     $$         120.0000      60.0000
  1.0    1      $$      S$2     $$         109.5000      60.0000
  1.0    1      $$      P$4     $$         109.5000     110.0000
  1.0    1      C$$     S$2     H$$         96.0000      44.0000
  1.0    1      C$$     S$2     C$$         99.0000      62.0000
  1.0    1      C$$     S$2     S$$         96.0000      44.0000
#torsion_3      amber
> E = SUM(n=1,3) { V(n) * [ 1 + cos(n*Phi - Phi0(n)) ] }

!Ver  Ref      I      J      K      L           V1      Phi0
V2      Phi0          V3      Phi0
!---- ---      ----   ----   ----   ----        -------   ------
-------  ------     -------  ------
 1.0    3       *      CB     CD     *           0.0000      0.0
5.3000   180.0       0.0000     0.0
 1.0    1       *      C      C2     *           0.0000      0.0
0.0000     0.0       0.0000   180.0
 1.0    1       *      C      CA     *           0.0000      0.0
5.3000   180.0       0.0000     0.0
 1.0    1       *      C      CB     *           0.0000      0.0
4.4000   180.0       0.0000     0.0
 1.0    1       *      C      CD     *           0.0000      0.0
5.3000   180.0       0.0000     0.0
 1.0    1       *      C      CH     *           0.0000      0.0
0.0000     0.0       0.0000     0.0
```

```
 1.0   1      *      C      CJ     *          0.0000     0.0
3.1000  180.0        0.0000     0.0
 1.0   1      *      C      CM     *          0.0000     0.0
3.1000  180.0        0.0000     0.0
 1.0   1      *      C      CT     *          0.0000     0.0
0.0000    0.0        0.0000     0.0
 1.0   1      *      C      N      *          0.0000     0.0
10.0000  180.0        0.0000     0.0
 1.0   1      *      C      N*     *          0.0000     0.0
5.8000  180.0        0.0000     0.0
 1.0   1      *      C      NA     *          0.0000     0.0
5.4000  180.0        0.0000     0.0
 1.0   1      *      C      NC     *          0.0000     0.0
8.0000  180.0        0.0000     0.0
 1.0   1      *      C      OH     *          0.0000     0.0
1.8000  180.0        0.0000     0.0
 1.0   1      *      C*     C2     *          0.0000     0.0
0.0000    0.0        0.0000     0.0
 1.0   1      *      C*     CB     *          0.0000     0.0
4.8000  180.0        0.0000     0.0
 1.0   1      *      C*     CG     *          0.0000     0.0
23.6000  180.0        0.0000     0.0
 1.0   1      *      C*     CT     *          0.0000     0.0
0.0000    0.0        0.0000     0.0
 1.0   1      *      C*     CW     *          0.0000     0.0
23.6000  180.0        0.0000     0.0
 1.0   1      *      C2     C2     *          0.0000     0.0
0.0000    0.0        2.0000     0.0
 1.0   1      *      C2     CA     *          0.0000     0.0
0.0000    0.0        0.0000     0.0
 1.0   1      *      C2     CC     *          0.0000     0.0
0.0000    0.0        0.0000     0.0
 1.0   1      *      C2     CH     *          0.0000     0.0
0.0000    0.0        2.0000     0.0
 1.0   1      *      C2     N      *          0.0000     0.0
0.0000    0.0        0.0000     0.0
 1.0   1      *      C2     N2     *          0.0000     0.0
0.0000    0.0        0.0000     0.0
 1.0   1      *      C2     N3     *          0.0000     0.0
```

```
0.0000    0.0      1.4000    0.0
 1.0   1      *     C2     NT     *          0.0000     0.0
0.0000    0.0      1.0000    0.0
 1.0   1      *     C2     OH     *          0.0000     0.0
0.0000    0.0      0.5000    0.0
 1.0   1      *     C2     OS     *          0.0000     0.0
0.0000    0.0      1.4500    0.0
 1.0   1      *     C2     S      *          0.0000     0.0
0.0000    0.0      1.0000    0.0
 1.0   1      *     C2     SH     *          0.0000     0.0
0.0000    0.0      0.7500    0.0
 1.0   1      *     CA     CA     *          0.0000     0.0
5.3000  180.0      0.0000    0.0
 1.0   1      *     CA     CB     *          0.0000     0.0
10.2000  180.0     0.0000    0.0
 1.0   1      *     CA     CD     *          0.0000     0.0
5.3000  180.0      0.0000    0.0
 1.0   1      *     CA     CJ     *          0.0000     0.0
3.7000  180.0      0.0000    0.0
 1.0   1      *     CA     CM     *          0.0000     0.0
3.7000  180.0      0.0000    0.0
 1.0   1      *     CA     CN     *          0.0000     0.0
10.6000  180.0     0.0000    0.0
 1.0   1      *     CA     CT     *          0.0000     0.0
0.0000    0.0      0.0000    0.0
 1.0   1      *     CA     N2     *          0.0000     0.0
6.8000  180.0      0.0000    0.0
 1.0   1      *     CA     NA     *          0.0000     0.0
6.0000  180.0      0.0000    0.0
 1.0   1      *     CA     NC     *          0.0000     0.0
9.6000  180.0      0.0000    0.0
 1.0   1      *     CB     CB     *          0.0000     0.0
16.3000  180.0     0.0000    0.0
 1.0   1      *     CB     CN     *          0.0000     0.0
20.0000  180.0     0.0000    0.0
 1.0   1      *     CB     N*     *          0.0000     0.0
6.6000  180.0      0.0000    0.0
 1.0   1      *     CB     NB     *          0.0000     0.0
5.1000  180.0      0.0000    0.0
```

```
 1.0    3       *       CB      NC      *               0.0000      0.0
8.3000   180.0          0.0000      0.0
 1.0    1       *       CC      CF      *               0.0000      0.0
14.3000  180.0          0.0000      0.0
 1.0    1       *       CC      CG      *               0.0000      0.0
15.9000  180.0          0.0000      0.0
 1.0    1       *       CC      CT      *               0.0000      0.0
0.0000     0.0          0.0000      0.0
 1.0    1       *       CC      CV      *               0.0000      0.0
14.3000  180.0          0.0000      0.0
 1.0    1       *       CC      CW      *               0.0000      0.0
15.9000  180.0          0.0000      0.0
 1.0    1       *       CC      NA      *               0.0000      0.0
5.6000   180.0          0.0000      0.0
 1.0    1       *       CC      NB      *               0.0000      0.0
4.8000   180.0          0.0000      0.0
 1.0    1       *       CD      CD      *               0.0000      0.0
5.3000   180.0          0.0000      0.0
 1.0    1       *       CD      CN      *               0.0000      0.0
5.3000   180.0          0.0000      0.0
 1.0    1       *       CE      N*      *               0.0000      0.0
6.7000   180.0          0.0000      0.0
 1.0    1       *       CE      NB      *               0.0000      0.0
20.0000  180.0          0.0000      0.0
 1.0    1       *       CF      NB      *               0.0000      0.0
4.8000   180.0          0.0000      0.0
 1.0    1       *       CG      NA      *               0.0000      0.0
6.0000   180.0          0.0000      0.0
 1.0    1       *       CH      CH      *               0.0000      0.0
0.0000     0.0          2.0000      0.0
 1.0    1       *       CH      N       *               0.0000      0.0
0.0000     0.0          0.0000      0.0
 1.0    1       *       CH      N*      *               0.0000      0.0
0.0000     0.0          0.0000      0.0
 1.0    1       *       CH      NT      *               0.0000      0.0
0.0000     0.0          1.0000      0.0
 1.0    1       *       CH      OH      *               0.0000      0.0
0.0000     0.0          0.5000      0.0
 1.0    1       *       CH      OS      *               0.0000      0.0
```

```
0.0000    0.0      1.4500    0.0
 1.0   1      *     CI    NC     *          0.0000    0.0
13.5000  180.0     0.0000    0.0
 1.0   1      *     CJ    CJ     *          0.0000    0.0
24.4000  180.0     0.0000    0.0
 1.0   1      *     CJ    CM     *          0.0000    0.0
24.4000  180.0     0.0000    0.0
 1.0   1      *     CJ    N*     *          0.0000    0.0
7.4000   180.0     0.0000    0.0
 1.0   1      *     CK    N*     *          0.0000    0.0
6.7000   180.0     0.0000    0.0
 1.0   1      *     CK    NB     *          0.0000    0.0
20.0000  180.0     0.0000    0.0
 1.0   1      *     CM    CM     *          0.0000    0.0
24.4000  180.0     0.0000    0.0
 1.0   1      *     CM    CT     *          0.0000    0.0
0.0000    0.0     0.0000    0.0
 1.0   1      *     CM    N*     *          0.0000    0.0
7.4000   180.0     0.0000    0.0
 1.0   1      *     CN    NA     *          0.0000    0.0
12.2000  180.0     0.0000    0.0
 1.0   1      *     CP    NA     *          0.0000    0.0
9.3000   180.0     0.0000    0.0
 1.0   1      *     CP    NB     *          0.0000    0.0
10.0000  180.0     0.0000    0.0
 1.0   1      *     CQ    NC     *          0.0000    0.0
13.5000  180.0     0.0000    0.0
 1.0   1      *     CR    NA     *          0.0000    0.0
9.3000   180.0     0.0000    0.0
 1.0   1      *     CR    NB     *          0.0000    0.0
10.0000  180.0     0.0000    0.0
 1.0   1      *     CT    CT     *          0.0000    0.0
0.0000    0.0     1.3000    0.0
 1.0   1      *     CT    N      *          0.0000    0.0
0.0000    0.0     0.0000    0.0
 1.0   1      *     CT    N*     *          0.0000    0.0
0.0000    0.0     0.0000    0.0
 1.0   1      *     CT    N2     *          0.0000    0.0
0.0000    0.0     0.0000    0.0
```

```
 1.0   1      *     CT     N3     *          0.0000     0.0
0.0000     0.0       1.4000     0.0
 1.0   1      *     CT     OH     *          0.0000     0.0
0.0000     0.0       0.5000     0.0
 1.0   1      *     CT     OS     *          0.0000     0.0
0.0000     0.0       1.1500     0.0
 1.0   1      *     CT     S      *          0.0000     0.0
0.0000     0.0       1.0000     0.0
 1.0   1      *     CT     SH     *          0.0000     0.0
0.0000     0.0       0.7500     0.0
 1.0   1      *     CV     NB     *          0.0000     0.0
4.8000   180.0       0.0000     0.0
 1.0   1      *     CW     NA     *          0.0000     0.0
6.0000   180.0       0.0000     0.0
 1.0   1      *     OH     P      *          0.0000     0.0
0.0000     0.0       0.7500     0.0
 1.0   1      *     OS     P      *          0.0000     0.0
0.0000     0.0       0.7500     0.0
 1.0   1      O     C      C2     N          0.0000     0.0
0.0000     0.0       0.2000   180.0
 1.0   1      O     C      CH     C2         0.0000     0.0
0.0000     0.0       0.1000   180.0
 1.0   1      O     C      CH     N          0.0000     0.0
0.0000     0.0       0.1000   180.0
 1.0   1      O     C      CH     CH         0.0000     0.0
0.0000     0.0       0.1000   180.0
 1.0   1     OS     C2     C2     OH         0.0000     0.0
0.5000     0.0       2.0000     0.0
 1.0   2     OH     C2     C2     OH         0.0000     0.0
0.5000     0.0       2.0000     0.0
 1.0   1     OS     C2     C2     OS         0.0000     0.0
0.5000     0.0       2.0000     0.0
 1.0   1     OS     C2     CH     OS         0.0000     0.0
0.5000     0.0       1.0000     0.0
 1.0   1     OS     C2     CH     OH         0.0000     0.0
0.5000     0.0       1.0000     0.0
 1.0   1     OH     C2     CH     OH         0.0000     0.0
0.5000     0.0       1.0000     0.0
 1.0   1     C2     C2     S      LP         0.0000     0.0
```

```
0.0000     0.0       0.0000     0.0
 1.0    1       CH     C2      SH      LP          0.0000     0.0
0.0000     0.0       0.0000     0.0
 1.0    1       OS     CH      C2      OH          0.0000     0.0
0.5000     0.0       1.0000     0.0
 1.0    1       OH     CH      CH      OH          0.0000     0.0
0.5000     0.0       0.5000     0.0
 1.0    1       OS     CH      CH      OH          0.0000     0.0
0.5000     0.0       0.5000     0.0
 1.0    1       OS     CH      CH      OS          0.0000     0.0
0.5000     0.0       0.5000     0.0
 1.0    1       HC     CM      CM      CT          0.0000     0.0
1.7100   180.0       0.0000     0.0
 1.0    1       C      CM      CM      HC          0.0000     0.0
6.5900   180.0       0.0000     0.0
 1.0    1       N*     CM      CM      CT          0.0000     0.0
6.5900   180.0       0.0000     0.0
 1.0    1       CA     CM      CM      HC          0.0000     0.0
6.5900   180.0       0.0000     0.0
 1.0    1       N*     CM      CM      CA          0.0000     0.0
9.5100   180.0       0.0000     0.0
 1.0    1       HC     CM      CM      HC          0.0000     0.0
1.7100   180.0       0.0000     0.0
 1.0    1       N*     CM      CM      C           0.0000     0.0
9.5100   180.0       0.0000     0.0
 1.0    1       N*     CM      CM      HC          0.0000     0.0
6.5900   180.0       0.0000     0.0
 1.0    1       N      CT      C       O           0.0000     0.0
0.0000     0.0       0.0670   180.0
 1.0    1       HC     CT      C       O           0.0000     0.0
0.0000     0.0       0.0670   180.0
 1.0    1       CT     CT      C       O           0.0000     0.0
0.0000     0.0       0.0670   180.0
 1.0    1       CT     OS      CT      CT          0.0000     0.0
0.2000   180.0       0.3830     0.0
 1.0    1       OS     CT      CT      OS          0.0000     0.0
0.5000     0.0       0.1440     0.0
 1.0    1       OS     CT      CT      OH          0.0000     0.0
0.5000     0.0       0.1440     0.0
```

1.0	1	OH	CT	CT	OH	0.0000	0.0
0.5000	0.0	0.1440	0.0				
1.0	1	H	N	C	O	0.6500	0.0
2.5000	180.0	0.0000	0.0				
1.0	1	C2	OS	C2	C3	0.0000	0.0
0.1000	0.0	0.7250	0.0				
1.0	1	C2	OS	C2	C2	0.0000	0.0
0.1000	0.0	1.4500	0.0				
1.0	1	C3	OS	C2	C3	0.0000	0.0
0.1000	0.0	1.4500	0.0				
1.0	1	CH	OS	CH	C2	0.0000	0.0
0.1000	0.0	0.7250	0.0				
1.0	1	CH	OS	CH	CH	0.0000	0.0
0.1000	0.0	0.7250	0.0				
1.0	1	C2	OS	CH	C2	0.0000	0.0
0.1000	0.0	0.7250	0.0				
1.0	1	C3	OS	CH	C3	0.0000	0.0
0.1000	0.0	0.7250	0.0				
1.0	1	CH	OS	CH	N*	0.0000	0.0
0.0000	0.0	0.7250	0.0				
1.0	1	C2	OS	CH	C3	0.0000	0.0
0.1000	0.0	0.7250	0.0				
1.0	1	OH	P	OS	C3	0.0000	0.0
0.7500	0.0	0.2500	0.0				
1.0	1	OS	P	OS	C2	0.0000	0.0
0.7500	0.0	0.2500	0.0				
1.0	1	OH	P	OS	C2	0.0000	0.0
0.7500	0.0	0.2500	0.0				
1.0	1	OS	P	OS	CT	0.0000	0.0
0.7500	0.0	0.2500	0.0				
1.0	1	OS	P	OS	CH	0.0000	0.0
0.7500	0.0	0.2500	0.0				
1.0	1	OS	P	OS	C3	0.0000	0.0
0.7500	0.0	0.2500	0.0				
1.0	1	OH	P	OS	CH	0.0000	0.0
0.7500	0.0	0.2500	0.0				
1.0	1	OH	P	OS	CT	0.0000	0.0
0.7500	0.0	0.2500	0.0				
1.0	1	LP	S	S	LP	0.0000	0.0

```
0.0000    0.0      0.0000    0.0
 1.0   1      LP    S      S      C2          0.0000    0.0
0.0000    0.0      0.0000    0.0
 1.0   1      C2    S      S      C2          0.0000    0.0
3.5000    0.0      0.6000    0.0
 1.0   1      CT    S      S      CT          0.0000    0.0
3.5000    0.0      0.6000    0.0
 1.0   1      LP    S      S      CT          0.0000    0.0
0.0000    0.0      0.0000    0.0
 1.1   4      *     CS     CS     *           0.0000    0.0
0.0000    0.0      1.0210    0.0
 1.1   4      *     CS     CT     *           0.0000    0.0
0.0000    0.0      1.0210    0.0
 1.1   4      *     AC     CS     *           0.0000    0.0
0.0000    0.0      1.0210    0.0
 1.1   4      *     BC     CS     *           0.0000    0.0
0.0000    0.0      1.0210    0.0
 1.1   4      *     CS     OT     *           0.0000    0.0
0.0000    0.0      0.4430    0.0
 1.1   4      *     CS     OE     *           0.0000    0.0
0.0000    0.0      0.9280    0.0
 1.1   4      *     AC     OE     *           0.0000    0.0
0.0000    0.0      0.9280    0.0
 1.1   4      *     BC     OE     *           0.0000    0.0
0.0000    0.0      0.9280    0.0
 1.1   4      *     AC     OA     *           0.0000    0.0
0.0000    0.0      0.0000    0.0
 1.1   4      *     BC     OB     *           0.0000    0.0
0.0000    0.0      0.0000    0.0
 1.1   4      *     CS     OA     *           0.0000    0.0
0.0000    0.0      0.0000    0.0
 1.1   4      *     CS     OB     *           0.0000    0.0
0.0000    0.0      0.0000    0.0
 1.1   4      *     CS     N      *           0.0000    0.0
0.0000    0.0      0.0000    0.0
 1.1   4      *     C      N      *           0.0000    0.0
10.0000  180.0      0.0000    0.0
 1.1   4      *     C      CS     *           0.0000    0.0
0.0000    0.0      0.0000    0.0
```

```
 1.1    4       OE      AC      OA      CS          2.1500   300.0
0.0000     0.0        0.0000     0.0
 1.1    4       AH      AC      OA      CS          0.0000     0.0
1.7500    60.0        0.0000     0.0
 1.1    4       CS      AC      OA      CS          0.0000     0.0
0.0000     0.0        0.8500     0.0
 1.1    4       OE      AC      OA      HY          2.1500   300.0
0.0000     0.0        0.0000     0.0
 1.1    4       AH      AC      OA      HY          0.0000     0.0
1.7500    60.0        0.0000     0.0
 1.1    4       CS      AC      OA      HY          0.0000     0.0
0.0000     0.0        0.8500     0.0
 1.1    4       OE      BC      OB      CS         -1.0500     0.0
0.0000     0.0        0.0000     0.0
 1.1    4       BH      BC      OB      CS          0.0000     0.0
1.2500   240.0        0.0000     0.0
 1.1    4       CS      BC      OB      CS          0.0000     0.0
0.0000     0.0        1.4000     0.0
 1.1    4       OE      BC      OB      HY         -1.0500     0.0
0.0000     0.0        0.0000     0.0
 1.1    4       BH      BC      OB      HY          0.0000     0.0
1.2500   240.0        0.0000     0.0
 1.1    4       CS      BC      OB      HY          0.0000     0.0
0.0000     0.0        1.4000     0.0
 1.1    4       HT      AC      OA      CS          0.0000     0.0
0.0000     0.0        0.8500     0.0
 1.1    4       HT      BC      OB      CS          0.0000     0.0
0.0000     0.0        1.4000     0.0
 1.1    4        H       N       C       O          0.6500     0.0
2.5000   180.0        0.0000     0.0
 1.1    4       HT      CS      C       O          0.0000     0.0
0.0000     0.0        0.0670   180.0
 1.0    1       $$      C$1     C$1     $$         0.0000     0.0
0.0000     0.0        1.3000     0.0
 1.0    1       $$      C$2     C$2     $$         0.0000     0.0
5.3000   180.0        0.0000     0.0
 1.0    1       $$      C$3     C$3     $$          0.0000     0.0
16.3000   180.0         0.0000     0.0
 1.0    1       $$      C$5     C$5     $$         0.0000     0.0
```

```
0.0000  180.0       0.0000     0.0
 1.0    1       $$     C$1    O$1    $$          0.0000     0.0
0.0000     0.0       1.1000     0.0
 1.0    1       $$     C$1    N$1    $$          0.0000     0.0
0.0000     0.0       0.3000     0.0
 1.0    1       $$     C$2    N$2    $$          0.0000     0.0
5.8000  180.0       0.0000     0.0
 1.0    1       $$     C$3    N$3    $$          0.0000     0.0
10.0000  180.0       0.0000     0.0
 1.0    1       $$     C$1    S$1    $$          0.0000     0.0
0.0000     0.0       0.7500     0.0
 1.0    1       $$     S$1    S$1    $$          0.0000     0.0
3.5000     0.0       0.6000     0.0
 1.0    1       $$     O$1    O$1    $$          0.0000     0.0
0.0000     0.0       1.1000     0.0
 1.0    1       $$     O$1    N$1    $$          0.0000     0.0
0.0000     0.0       1.1000     0.0
 1.0    1       $$     O$1    P$1    $$          0.0000     0.0
0.0000     0.0       0.7500     0.0
 1.0    1       $$     N$1    N$1    $$          0.0000     0.0
0.0000     0.0       0.3000     0.0
#out_of_plane  amber
> E = Kchi * [ 1 + cos(n*Chi - Chi0) ]
!Ver  Ref      I      J      K      L            Kchi         n
    Chi0
!---- ---     ----   ----   ----   ----       -------       ------
 -------
 1.0   3      C*     NA     CA     CA          0.0000         2
180.0000
 1.0   1      N3     C      CH     C2          7.0000         3
180.0000
 1.0   1      C3     CA     CH     C3          7.0000         3
180.0000
 1.0   1      C      NT     CH     C3         14.0000         3
180.0000
 1.0   1      N3     C      CH     CH          7.0000         3
180.0000
 1.0   1      H2     N2     CH     H2          0.0000         3
180.0000
```

```
 1.0   1      *      CH     C2     *          14.0000          3
180.0000
 1.0   1      *      CH     CH     *          14.0000          3
180.0000
 1.0   1      *      CC     CC     *           0.0000          2
180.0000
 1.0   1      *      CC     CB     *           0.0000          2
180.0000
 1.0   1      C      N      CH     *          14.0000          3
180.0000
 1.0   1      C2     N      CH     *           1.0000          2
180.0000
 1.0   1      CT     N      CT     *           1.0000          2
180.0000
 1.0   1      H2     N      H2     *           1.0000          2
180.0000
 1.0   1      N2     CA     N2     *          10.5000          2
180.0000
 1.0   1      O2     C      O2     *          10.5000          2
180.0000
 1.0   1      C      NT     CH     *          14.0000          3
180.0000
 1.0   1      C      N3     CH     *          14.0000          3
180.0000
 1.0   1      O      C      *      *          10.5000          2
180.0000
 1.0   1      HC     C*     *      *           0.0000          2
180.0000
 1.0   1      HC     CW     *      *           0.0000          2
180.0000
 1.0   1      CB     CN     *      *           0.0000          2
180.0000
 1.0   1      CN     CB     *      *           0.0000          2
180.0000
 1.0   1      C*     CB     *      *           0.0000          2
180.0000
 1.0   1      CA     CB     *      *           0.0000          2
180.0000
 1.0   1      CA     CN     *      *           0.0000          2
```

```
180.0000
 1.0    1      NA     CN     *      *          0.0000          2
180.0000
 1.0    1      HC     CA     *      *          2.0000          2
180.0000
 1.0    1      H      N      *      *          1.0000          2
180.0000
 1.0    1      H2     N2     *      *          1.0000          2
180.0000
 1.0    1      H3     N2     *      *          1.0000          2
180.0000
 1.0    1      H2     NT     *      *          1.0000          2
180.0000
 1.0    1      H      NA     *      *          1.0000          2
180.0000
 1.0    1      $$     $$     $$     $$        10.0000          2
180.0000
#nonbond(12-6) amber
@type r-eps
@combination arithmetic
> E = EPSij * { (Rij*/Rij)^12 - 2(Rij*/Rij)^6 }
> where  EPSij = sqrt( EPSi * EPSj)
>          Rij* = (Ri* + Rj*)/2
!Ver  Ref      I              Ri*            EPSi
!---- ---     ----        -----------    -----------
 1.0    3      IM            5.0000        0.10000
 1.0    3      CU            2.4000        0.05000
 1.0    3      I             4.8000        0.40000
 1.0    3      OW            3.5360        0.15200
 1.0    3      MG            2.3400        0.10000
 1.0    3      C0            3.2000        0.10000
 1.0    3      QC            6.8000        0.00008
 1.0    3      QK            5.3200        0.00033
 1.0    3      QL            2.2800        0.01800
 1.0    3      QN            3.7400        0.00280
 1.0    3      QR            5.9200        0.00017
 1.0    1      C             3.7000        0.12000
 1.0    1      C*            3.7000        0.12000
 1.0    1      C2            3.8400        0.12000
```

1.0	1	C3	4.0000	0.15000
1.0	1	CA	3.7000	0.12000
1.0	1	CB	3.7000	0.12000
1.0	1	CC	3.7000	0.12000
1.0	1	CD	3.7000	0.12000
1.0	1	CE	3.7000	0.12000
1.0	1	CF	3.7000	0.12000
1.0	1	CG	3.7000	0.12000
1.0	1	CH	3.7000	0.09000
1.0	1	CI	3.7000	0.12000
1.0	1	CJ	3.7000	0.12000
1.0	1	CK	3.7000	0.12000
1.0	1	CM	3.7000	0.12000
1.0	1	CN	3.7000	0.12000
1.0	1	CP	3.7000	0.12000
1.0	1	CQ	3.7000	0.12000
1.0	1	CR	3.7000	0.12000
1.0	1	CT	3.6000	0.06000
1.0	1	CV	3.7000	0.12000
1.0	1	CW	3.7000	0.12000
1.0	1	H	2.0000	0.02000
1.0	1	H2	2.0000	0.02000
1.0	1	H3	2.0000	0.02000
1.0	1	HC	3.0800	0.01000
1.0	1	HO	2.0000	0.02000
1.0	1	HS	2.0000	0.02000
1.0	1	LP	2.4000	0.01600
1.0	1	N	3.5000	0.16000
1.0	1	N*	3.5000	0.16000
1.0	1	N2	3.5000	0.16000
1.0	1	N3	3.7000	0.08000
1.0	1	NA	3.5000	0.16000
1.0	1	NB	3.5000	0.16000
1.0	1	NC	3.5000	0.16000
1.0	1	NP	3.5000	0.16000
1.0	1	NT	3.7000	0.12000
1.0	1	O	3.2000	0.20000
1.0	1	O2	3.2000	0.20000
1.0	1	OH	3.3000	0.15000

```
 1.0    1      OS              3.3000         0.15000
 1.0    1      P               4.2000         0.20000
 1.0    1      S               4.0000         0.20000
 1.0    1      SH              4.0000         0.20000
 1.1    4      CS              3.6000         0.09030
 1.1    4      AC              3.6000         0.09030
 1.1    4      BC              3.6000         0.09030
 1.1    4      C               3.7000         0.12000
 1.1    4      H               2.0000         0.02000
 1.1    4      HY              1.6000         0.04980
 1.1    4      HT              2.9360         0.00450
 1.1    4      HO              2.0000         0.02000
 1.1    4      AH              2.9360         0.00450
 1.1    4      BH              2.9360         0.00450
 1.1    4      OT              3.2000         0.15910
 1.1    4      OA              3.2000         0.15910
 1.1    4      OB              3.2000         0.15910
 1.1    4      OE              3.2000         0.15910
 1.1    4      OH              3.3000         0.15000
 1.1    4      O               3.2000         0.20000
 1.1    4      N               3.5000         0.16000
#hydrogen_bond(10-12)     amber
> E = Aij/r^12 - Bij/r^10
!Ver   Ref      I     J           A              B
!----- ---     ----- ----     -----------    -----------
 1.0    3      H     OS          7557.0000      2385.0000
 1.0    3      H     OW          7557.0000      2385.0000
 1.0    3      H2    OS          7557.0000      2385.0000
 1.0    3      H2    OW          7557.0000      2385.0000
 1.0    3      HW    NB          7557.0000      2385.0000
 1.0    3      HW    NC         10238.0000      3071.0000
 1.0    3      HW    O           7557.0000      2385.0000
 1.0    3      HW    O2          4019.0000      1409.0000
 1.0    3      HW    OH          7557.0000      2385.0000
 1.0    3      HW    OS          7557.0000      2385.0000
 1.0    3      HW    S         265720.0000     35429.0000
 1.0    3      HW    SH        265720.0000     35429.0000
 1.0    1      H     NB          7557.0000      2385.0000
 1.0    1      H     NC         10238.0000      3071.0000
```

```
 1.0    1      H     O2        4019.0000     1409.0000
 1.0    1      H     O         7557.0000     2385.0000
 1.0    1      H     OH        7557.0000     2385.0000
 1.0    3      H     S       265720.0000    35429.0000
 1.0    3      H     SH      265720.0000    35429.0000
 1.0    1      HO    NB        7557.0000     2385.0000
 1.0    1      HO    NC        7557.0000     2385.0000
 1.0    1      HO    O2        4019.0000     1409.0000
 1.0    1      HO    O         7557.0000     2385.0000
 1.0    1      HO    OH        7557.0000     2385.0000
 1.0    3      HO    S       265720.0000    35429.0000
 1.0    3      HO    SH      265720.0000    35429.0000
 1.0    1      H2    NB        4019.0000     1409.0000
 1.0    1      H2    NC        4019.0000     1409.0000
 1.0    1      H2    O2        4019.0000     1409.0000
 1.0    1      H2    O        10238.0000     3071.0000
 1.0    1      H2    OH        4019.0000     1409.0000
 1.0    3      H2    S       265720.0000    35429.0000
 1.0    3      H2    SH      265720.0000    35429.0000
 1.0    1      H3    NB        4019.0000     1409.0000
 1.0    1      H3    NC        4019.0000     1409.0000
 1.0    1      H3    O2        4019.0000     1409.0000
 1.0    1      H3    O         7557.0000     2385.0000
 1.0    1      H3    OH        7557.0000     2385.0000
 1.0    3      H3    S       265720.0000    35429.0000
 1.0    3      H3    SH      265720.0000    35429.0000
 1.0    1      HS    NB       14184.0000     3082.0000
 1.0    1      HS    NC       14184.0000     3082.0000
 1.0    1      HS    O2       14184.0000     3082.0000
 1.0    1      HS    O        14184.0000     3082.0000
 1.0    1      HS    OH       14184.0000     3082.0000
 1.0    3      HS    S       265720.0000    35429.0000
 1.0    3      HS    SH      265720.0000    35429.0000
#bond_increments    amber
!Ver  Ref     I     J       DeltaIJ      DeltaJI
!---- ---    ----  ----    -------      -------
 1.1   5      CM    CM        0.000        0.000
 1.1   5      CA    CA        0.000        0.000
 1.1   5      CB    CB        0.000        0.000
```

1.1	5	C5	C6	0.000	0.000
1.1	5	CT	CT	0.000	0.000
1.1	5	HT	CT	0.066	-0.066
1.1	5	H	NT	0.133	-0.133
1.1	5	NT	CT	-0.189	0.189
1.1	5	CA	OH	0.334	-0.334
1.1	5	CT	OS	0.237	-0.237
1.1	5	HC	CT	0.066	-0.066
1.1	6	CS	CS	0.000	0.000
1.1	6	AC	CS	0.000	0.000
1.1	6	BC	CS	0.000	0.000
1.1	6	CS	CT	0.000	0.000
1.1	6	CS	OS	0.200	-0.200
1.1	5	N*	CS	-0.183	0.183
1.1	6	OT	HY	-0.400	0.400
1.1	6	OA	HY	-0.400	0.400
1.1	6	OB	HY	-0.400	0.400
1.1	6	CS	HT	-0.100	0.100
1.1	5	AC	AH	-0.100	0.100
1.1	6	BC	BH	-0.100	0.100
1.1	6	AC	HT	-0.100	0.100
1.1	6	BC	HT	-0.100	0.100
1.1	6	AC	CA	0.250	-0.250
1.1	6	BC	OB	0.250	-0.250
1.1	6	CS	OA	0.250	-0.250
1.1	6	CS	OB	0.250	-0.250
1.1	6	CS	OT	0.250	-0.250
1.1	6	CS	OE	0.200	-0.200
1.1	6	AC	OE	0.200	-0.200
1.1	5	BC	OE	0.200	-0.200
1.1	6	OW	HW	-0.380	0.380
1.1	5	N*	CT	-0.183	0.183
1.1	5	P	OS	0.254	-0.254
1.1	5	CB	N*	0.130	-0.130
1.1	5	CK	N*	-0.253	0.253
1.1	5	NC	CB	-0.335	0.335
1.1	5	NB	CB	0.020	-0.020
1.1	5	CB	CA	0.000	-0.000
1.1	5	CK	NB	0.566	-0.566

1.1	5	CK	HC	-0.051	0.051
1.1	5	N2	CA	-0.162	0.162
1.1	5	NC	CA	-0.430	0.430
1.1	5	H2	N2	0.318	-0.318
1.1	5	CQ	NC	0.341	-0.341
1.1	5	CQ	HC	0.005	-0.005
1.1	5	O2	P	-0.913	0.413
1.1	5	C	N*	-0.044	0.044
1.1	5	CM	N*	0.137	-0.137
1.1	5	NA	C	-0.255	0.255
1.1	5	O	C	-0.492	0.492
1.1	5	NA	H	-0.282	0.282
1.1	5	CM	C	-0.150	0.150
1.1	5	CM	CT	0.055	-0.055
1.1	5	CM	HC	-0.101	0.101
1.1	5	H2	CT	0.119	-0.119
1.1	5	C	NC	0.424	-0.424
1.1	5	CM	CA	-0.409	0.409
1.1	5	N2	HC	-0.037	0.037
1.1	5	OH	CT	-0.263	0.263
1.1	5	HO	OH	0.303	-0.303
1.1	5	C	CB	-0.005	0.005
1.1	5	NA	CA	-0.215	0.215
1.1	5	CT	N	0.171	-0.171
1.1	5	H	N	0.274	-0.274
1.1	5	C	CT	0.095	-0.095
1.1	5	C	N	0.139	-0.139
1.1	5	N2	CT	0.044	-0.044
1.1	5	H3	N2	0.551	-0.351
1.1	5	O2	C	-0.792	0.292
1.1	5	S	CT	-0.023	0.023
1.1	5	LP	S	-0.403	0.403
1.1	5	SH	CT	-0.033	0.033
1.1	5	HS	SH	0.127	-0.127
1.1	5	SH	LP	0.489	-0.489
1.1	5	CC	CT	0.007	-0.007
1.1	5	NB	CC	-0.256	0.256
1.1	5	CW	CC	0.018	-0.018
1.1	5	CR	NB	0.251	-0.251

1.1	5	NA	CR	-0.066	0.066
1.1	5	CR	HC	-0.067	0.067
1.1	5	CW	NA	-0.057	0.057
1.1	5	CW	HC	-0.099	0.099
1.1	5	NA	CC	-0.020	0.020
1.1	5	NA	PS	0.423	-0.423
1.1	5	CV	CC	0.035	-0.035
1.1	5	CV	NB	0.227	-0.227
1.1	5	CV	HC	-0.042	0.042
1.1	5	N3	CT	0.905	0.095
1.1	5	N3	H3	-0.326	0.326
1.1	5	CA	CT	-0.033	0.033
1.1	5	CA	HC	-0.101	0.101
1.1	5	C*	CT	0.005	-0.005
1.1	5	C*	CW	-0.192	0.192
1.1	5	CB	C*	-0.045	0.045
1.1	5	CN	NA	0.176	-0.176
1.1	5	CN	CA	0.074	-0.074
1.1	5	CB	CN	0.104	-0.104
1.1	5	CA	C	-0.181	0.181
1.1	5	OH	C	-0.081	0.081

#reference 1
creation of file
#reference 2
Lone pair lp had incorrect mass of 0.001097.
Angle CT-C-O2 was by error included twice.
Torsion OH-C2-C2-OH was written as two separate lines.
Hence only one of the energy terms was included.
@Author Jon Hurley
@Date 13-December-90
#reference 3
parameter set modified with the addtional parameters
from kollman's parm89a rev a force field file
note that the HW...OW hydrogen bond parameters and
the HW van der waals parameters are not included in
the files since they are equal to zero in parm89a.
@Author tom thacher
@Date 11-March-92
#reference 4

```
homans' carbohydrate potential
@Author Tom Thacher
@Date 7-July-1992
#reference 5
bond increments
@Author Tom Thacher
@Date 7-July-1992
#end

************************************************************
************************************************************
                        END OF LISTING
************************************************************
************************************************************

************************************************************
          DATA FILE FOR H BOND FORCES - HBOND.DAT
************************************************************

47 !data items
!BIOSYM forcefield          2
!version amber.frc  1.0  19-Oct-90
!version amber.frc  1.1   8-Aug-92
!define amber
! This is the new format version of the amber forcefield
!hbond_definition       amber
!1.0   1   distance      2.5000
!1.0   1   angle        90.0000
!1.0   1   donors        H   HO  H2  H3  HS
!1.0   1   acceptors     NB  NC  O2  O   OH  S   SH
!hydrogen_bond(10-12)    amber
! E = Aij/r^12 - Bij/r^10
!Ver  Ref     I    J          A             B
!---- ---    ---- ----    -----------   -----------
 1.0   3     H    OS        7557.0000     2385.0000
 1.0   3     H    OW        7557.0000     2385.0000
 1.0   3     H2   OS        7557.0000     2385.0000
 1.0   3     H2   OW        7557.0000     2385.0000
 1.0   3     HW   NB        7557.0000     2385.0000
```

1.0	3	HW	NC	10238.0000	3071.0000
1.0	3	HW	O	7557.0000	2385.0000
1.0	3	HW	O2	4019.0000	1409.0000
1.0	3	HW	OH	7557.0000	2385.0000
1.0	3	HW	OS	7557.0000	2385.0000
1.0	3	HW	S	265720.0000	35429.0000
1.0	3	HW	SH	265720.0000	35429.0000
1.0	1	H	NB	7557.0000	2385.0000
1.0	1	H	NC	10238.0000	3071.0000
1.0	1	H	O2	4019.0000	1409.0000
1.0	1	H	O	7557.0000	2385.0000
1.0	1	H	OH	7557.0000	2385.0000
1.0	3	H	S	265720.0000	35429.0000
1.0	3	H	SH	265720.0000	35429.0000
1.0	1	HO	NB	7557.0000	2385.0000
1.0	1	HO	NC	7557.0000	2385.0000
1.0	1	HO	O2	4019.0000	1409.0000
1.0	1	HO	O	7557.0000	2385.0000
1.0	1	HO	OH	7557.0000	2385.0000
1.0	3	HO	S	265720.0000	35429.0000
1.0	3	HO	SH	265720.0000	35429.0000
1.0	1	H2	NB	4019.0000	1409.0000
1.0	1	H2	NC	4019.0000	1409.0000
1.0	1	H2	O2	4019.0000	1409.0000
1.0	1	H2	O	10238.0000	3071.0000
1.0	1	H2	OH	4019.0000	1409.0000
1.0	3	H2	S	265720.0000	35429.0000
1.0	3	H2	SH	265720.0000	35429.0000
1.0	1	H3	NB	4019.0000	1409.0000
1.0	1	H3	NC	4019.0000	1409.0000
1.0	1	H3	O2	4019.0000	1409.0000
1.0	1	H3	O	7557.0000	2385.0000
1.0	1	H3	OH	7557.0000	2385.0000
1.0	3	H3	S	265720.0000	35429.0000
1.0	3	H3	SH	265720.0000	35429.0000
1.0	1	HS	NB	14184.0000	3082.0000
1.0	1	HS	NC	14184.0000	3082.0000
1.0	1	HS	O2	14184.0000	3082.0000
1.0	1	HS	O	14184.0000	3082.0000

```
 1.0   1     HS   OH       14184.0000     3082.0000
 1.0   3     HS   S       265720.0000    35429.0000
 1.0   3     HS   SH      265720.0000    35429.0000

*****************************************************************
      DATA FILE FOR LENNARD JONES FORCES - LJ_PARAM.DAT
*****************************************************************

74 !total atoms
!BIOSYM forcefield          2
!version amber.frc  1.0  19-Oct-90
!version amber.frc  1.1   8-Aug-92
!define amber
! This is the new format version of the amber forcefield
!nonbond(12-6) amber
!type r-eps
!combination arithmetic
! E = EPSij * { (Rij*/Rij)^12 - 2(Rij*/Rij)^6 }
! where  EPSij = sqrt( EPSi * EPSj)
!         Rij* = (Ri* + Rj*)/2
!Ver  Ref     I            Ri*           EPSi
!---- ---     ----       ----------   ----------
 1.0   3     IM           5.0000       0.10000
 1.0   3     CU           2.4000       0.05000
 1.0   3     I            4.8000       0.40000
 1.0   3     OW           3.5360       0.15200
 1.0   3     MG           2.3400       0.10000
 1.0   3     C0           3.2000       0.10000
 1.0   3     QC           6.8000       0.00008
 1.0   3     QK           5.3200       0.00033
 1.0   3     QL           2.2800       0.01800
 1.0   3     QN           3.7400       0.00280
 1.0   3     QR           5.9200       0.00017
 1.0   1     C            3.7000       0.12000
 1.0   1     C*           3.7000       0.12000
 1.0   1     C2           3.8400       0.12000
 1.0   1     C3           4.0000       0.15000
 1.0   1     CA           3.7000       0.12000
 1.0   1     CB           3.7000       0.12000
```

1.0	1	CC	3.7000	0.12000
1.0	1	CD	3.7000	0.12000
1.0	1	CE	3.7000	0.12000
1.0	1	CF	3.7000	0.12000
1.0	1	CG	3.7000	0.12000
1.0	1	CH	3.7000	0.09000
1.0	1	CI	3.7000	0.12000
1.0	1	CJ	3.7000	0.12000
1.0	1	CK	3.7000	0.12000
1.0	1	CM	3.7000	0.12000
1.0	1	CN	3.7000	0.12000
1.0	1	CP	3.7000	0.12000
1.0	1	CQ	3.7000	0.12000
1.0	1	CR	3.7000	0.12000
1.0	1	CT	3.6000	0.06000
1.0	1	CV	3.7000	0.12000
1.0	1	CW	3.7000	0.12000
1.0	1	H	2.0000	0.02000
1.0	1	H2	2.0000	0.02000
1.0	1	H3	2.0000	0.02000
1.0	1	HC	3.0800	0.01000
1.0	1	HO	2.0000	0.02000
1.0	1	HS	2.0000	0.02000
1.0	1	LP	2.4000	0.01600
1.0	1	N	3.5000	0.16000
1.0	1	N*	3.5000	0.16000
1.0	1	N2	3.5000	0.16000
1.0	1	N3	3.7000	0.08000
1.0	1	NA	3.5000	0.16000
1.0	1	NB	3.5000	0.16000
1.0	1	NC	3.5000	0.16000
1.0	1	NP	3.5000	0.16000
1.0	1	NT	3.7000	0.12000
1.0	1	O	3.2000	0.20000
1.0	1	O2	3.2000	0.20000
1.0	1	OH	3.3000	0.15000
1.0	1	OS	3.3000	0.15000
1.0	1	P	4.2000	0.20000
1.0	1	S	4.0000	0.20000

```
 1.0    1      SH              4.0000        0.20000
 1.1    4      CS              3.6000        0.09030
 1.1    4      AC              3.6000        0.09030
 1.1    4      BC              3.6000        0.09030
 1.1    4      C               3.7000        0.12000
 1.1    4      H               2.0000        0.02000
 1.1    4      HY              1.6000        0.04980
 1.1    4      HT              2.9360        0.00450
 1.1    4      HO              2.0000        0.02000
 1.1    4      AH              2.9360        0.00450
 1.1    4      BH              2.9360        0.00450
 1.1    4      OT              3.2000        0.15910
 1.1    4      OA              3.2000        0.15910
 1.1    4      OB              3.2000        0.15910
 1.1    4      OE              3.2000        0.15910
 1.1    4      OH              3.3000        0.15000
 1.1    4      O               3.2000        0.20000
 1.1    4      N               3.5000        0.16000

*************************************************************
          DATA FILE FOR TORSION FORCES - TORSION.DAT
*************************************************************

179 ! total entries in this data file
!BIOSYM forcefield          2
!version amber.frc  1.0  19-Oct-90
!version amber.frc  1.1   8-Aug-92
!define amber
! This is the new format version of the amber forcefield
!torsion_3      amber
! E = SUM(n=1,3) { V(n) * [ 1 + cos(n*Phi - Phi0(n)) ] }
!Ver  Ref      I      J      K      L            V1      Phi0
V2      Phi0         V3      Phi0
!---- ---      ----   ----   ----   ----        -------   ------
-------  ------     -------  -----
 1.0    1      O      C      C2     N            0.0000     0.0
0.0000    0.0       0.2000  180.0
 1.0    1      O      C      CH     C2           0.0000     0.0
0.0000    0.0       0.1000  180.0
```

```
 1.0   1      O      C      CH     N          0.0000     0.0
0.0000    0.0       0.1000  180.0
 1.0   1      O      C      CH     CH         0.0000     0.0
0.0000    0.0       0.1000  180.0
 1.0   1      OS     C2     C2     OH         0.0000     0.0
0.5000    0.0       2.0000    0.0
 1.0   2      OH     C2     C2     OH         0.0000     0.0
0.5000    0.0       2.0000    0.0
 1.0   1      OS     C2     C2     OS         0.0000     0.0
0.5000    0.0       2.0000    0.0
 1.0   1      OS     C2     CH     OS         0.0000     0.0
0.5000    0.0       1.0000    0.0
 1.0   1      OS     C2     CH     OH         0.0000     0.0
0.5000    0.0       1.0000    0.0
 1.0   1      OH     C2     CH     OH         0.0000     0.0
0.5000    0.0       1.0000    0.0
 1.0   1      C2     C2     S      LP         0.0000     0.0
0.0000    0.0       0.0000    0.0
 1.0   1      CH     C2     SH     LP         0.0000     0.0
0.0000    0.0       0.0000    0.0
 1.0   1      OS     CH     C2     OH         0.0000     0.0
0.5000    0.0       1.0000    0.0
 1.0   1      OH     CH     CH     OH         0.0000     0.0
0.5000    0.0       0.5000    0.0
 1.0   1      OS     CH     CH     OH         0.0000     0.0
0.5000    0.0       0.5000    0.0
 1.0   1      OS     CH     CH     OS         0.0000     0.0
0.5000    0.0       0.5000    0.0
 1.0   1      HC     CM     CM     CT         0.0000     0.0
1.7100  180.0       0.0000    0.0
 1.0   1      C      CM     CM     HC         0.0000     0.0
6.5900  180.0       0.0000    0.0
 1.0   1      N*     CM     CM     CT         0.0000     0.0
6.5900  180.0       0.0000    0.0
 1.0   1      CA     CM     CM     HC         0.0000     0.0
6.5900  180.0       0.0000    0.0
 1.0   1      N*     CM     CM     CA         0.0000     0.0
9.5100  180.0       0.0000    0.0
 1.0   1      HC     CM     CM     HC         0.0000     0.0
```

```
1.7100  180.0       0.0000    0.0
 1.0    1      N*    CM      CM      C              0.0000    0.0
9.5100  180.0       0.0000    0.0
 1.0    1      N*    CM      CM      HC             0.0000    0.0
6.5900  180.0       0.0000    0.0
 1.0    1      N     CT      C       O              0.0000    0.0
0.0000    0.0       0.0670  180.0
 1.0    1      HC    CT      C       O              0.0000    0.0
0.0000    0.0       0.0670  180.0
 1.0    1      CT    CT      C       O              0.0000    0.0
0.0000    0.0       0.0670  180.0
 1.0    1      CT    OS      CT      CT             0.0000    0.0
0.2000  180.0       0.3830    0.0
 1.0    1      OS    CT      CT      OS             0.0000    0.0
0.5000    0.0       0.1440    0.0
 1.0    1      OS    CT      CT      OH             0.0000    0.0
0.5000    0.0       0.1440    0.0
 1.0    1      OH    CT      CT      OH             0.0000    0.0
0.5000    0.0       0.1440    0.0
 1.0    1      H     N       C       O              0.6500    0.0
2.5000  180.0       0.0000    0.0
 1.0    1      C2    OS      C2      C3             0.0000    0.0
0.1000    0.0       0.7250    0.0
 1.0    1      C2    OS      C2      C2             0.0000    0.0
0.1000    0.0       1.4500    0.0
 1.0    1      C3    OS      C2      C3             0.0000    0.0
0.1000    0.0       1.4500    0.0
 1.0    1      CH    OS      CH      C2             0.0000    0.0
0.1000    0.0       0.7250    0.0
 1.0    1      CH    OS      CH      CH             0.0000    0.0
0.1000    0.0       0.7250    0.0
 1.0    1      C2    OS      CH      C2             0.0000    0.0
0.1000    0.0       0.7250    0.0
 1.0    1      C3    OS      CH      C3             0.0000    0.0
0.1000    0.0       0.7250    0.0
 1.0    1      CH    OS      CH      N*             0.0000    0.0
0.0000    0.0       0.7250    0.0
 1.0    1      C2    OS      CH      C3             0.0000    0.0
0.1000    0.0       0.7250    0.0
```

```
 1.0    1       OH      P       OS      C3          0.0000      0.0
0.7500     0.0         0.2500     0.0
 1.0    1       OS      P       OS      C2          0.0000      0.0
0.7500     0.0         0.2500     0.0
 1.0    1       OH      P       OS      C2          0.0000      0.0
0.7500     0.0         0.2500     0.0
 1.0    1       OS      P       OS      CT          0.0000      0.0
0.7500     0.0         0.2500     0.0
 1.0    1       OS      P       OS      CH          0.0000      0.0
0.7500     0.0         0.2500     0.0
 1.0    1       OS      P       OS      C3          0.0000      0.0
0.7500     0.0         0.2500     0.0
 1.0    1       OH      P       OS      CH          0.0000      0.0
0.7500     0.0         0.2500     0.0
 1.0    1       OH      P       OS      CT          0.0000      0.0
0.7500     0.0         0.2500     0.0
 1.0    1       LP      S       S       LP          0.0000      0.0
0.0000     0.0         0.0000     0.0
 1.0    1       LP      S       S       C2          0.0000      0.0
0.0000     0.0         0.0000     0.0
 1.0    1       C2      S       S       C2          0.0000      0.0
3.5000     0.0         0.6000     0.0
 1.0    1       CT      S       S       CT          0.0000      0.0
3.5000     0.0         0.6000     0.0
 1.0    1       LP      S       S       CT          0.0000      0.0
0.0000     0.0         0.0000     0.0
 1.1    4       OE      AC      OA      CS          2.1500    300.0
0.0000     0.0         0.0000     0.0
 1.1    4       AH      AC      OA      CS          0.0000      0.0
1.7500    60.0         0.0000     0.0
 1.1    4       CS      AC      OA      CS          0.0000      0.0
0.0000     0.0         0.8500     0.0
 1.1    4       OE      AC      OA      HY          2.1500    300.0
0.0000     0.0         0.0000     0.0
 1.1    4       AH      AC      OA      HY          0.0000      0.0
1.7500    60.0         0.0000     0.0
 1.1    4       CS      AC      OA      HY          0.0000      0.0
0.0000     0.0         0.8500     0.0
 1.1    4       OE      BC      OB      CS         -1.0500      0.0
```

```
0.0000    0.0      0.0000    0.0
 1.1   4      BH    BC     OB     CS          0.0000    0.0
1.2500  240.0      0.0000    0.0
 1.1   4      CS    BC     OB     CS          0.0000    0.0
0.0000    0.0      1.4000    0.0
 1.1   4      OE    BC     OB     HY         -1.0500    0.0
0.0000    0.0      0.0000    0.0
 1.1   4      BH    BC     OB     HY          0.0000    0.0
1.2500  240.0      0.0000    0.0
 1.1   4      CS    BC     OB     HY          0.0000    0.0
0.0000    0.0      1.4000    0.0
 1.1   4      HT    AC     OA     CS          0.0000    0.0
0.0000    0.0      0.8500    0.0
 1.1   4      HT    BC     OB     CS          0.0000    0.0
0.0000    0.0      1.4000    0.0
 1.1   4      H     N      C      O           0.6500    0.0
2.5000  180.0      0.0000    0.0
 1.1   4      HT    CS     C      O           0.0000    0.0
0.0000    0.0      0.0670  180.0
 1.0   3      *     CB     CD     *           0.0000    0.0
5.3000  180.0      0.0000    0.0
 1.0   1      *     C      C2     *           0.0000    0.0
0.0000    0.0      0.0000  180.0
 1.0   1      *     C      CA     *           0.0000    0.0
5.3000  180.0      0.0000    0.0
 1.0   1      *     C      CB     *           0.0000    0.0
4.4000  180.0      0.0000    0.0
 1.0   1      *     C      CD     *           0.0000    0.0
5.3000  180.0      0.0000    0.0
 1.0   1      *     C      CH     *           0.0000    0.0
0.0000    0.0      0.0000    0.0
 1.0   1      *     C      CJ     *           0.0000    0.0
3.1000  180.0      0.0000    0.0
 1.0   1      *     C      CM     *           0.0000    0.0
3.1000  180.0      0.0000    0.0
 1.0   1      *     C      CT     *           0.0000    0.0
0.0000    0.0      0.0000    0.0
 1.0   1      *     C      N      *           0.0000    0.0
10.0000  180.0      0.0000    0.0
```

```
 1.0   1      *      C      N*     *          0.0000     0.0
5.8000  180.0       0.0000    0.0
 1.0   1      *      C      NA     *          0.0000     0.0
5.4000  180.0       0.0000    0.0
 1.0   1      *      C      NC     *          0.0000     0.0
8.0000  180.0       0.0000    0.0
 1.0   1      *      C      OH     *          0.0000     0.0
1.8000  180.0       0.0000    0.0
 1.0   1      *      C*     C2     *          0.0000     0.0
0.0000    0.0       0.0000    0.0
 1.0   1      *      C*     CB     *          0.0000     0.0
4.8000  180.0       0.0000    0.0
 1.0   1      *      C*     CG     *          0.0000     0.0
23.6000  180.0       0.0000    0.0
 1.0   1      *      C*     CT     *          0.0000     0.0
0.0000    0.0       0.0000    0.0
 1.0   1      *      C*     CW     *          0.0000     0.0
23.6000  180.0       0.0000    0.0
 1.0   1      *      C2     C2     *          0.0000     0.0
0.0000    0.0       2.0000    0.0
 1.0   1      *      C2     CA     *          0.0000     0.0
0.0000    0.0       0.0000    0.0
 1.0   1      *      C2     CC     *          0.0000     0.0
0.0000    0.0       0.0000    0.0
 1.0   1      *      C2     CH     *          0.0000     0.0
0.0000    0.0       2.0000    0.0
 1.0   1      *      C2     N      *          0.0000     0.0
0.0000    0.0       0.0000    0.0
 1.0   1      *      C2     N2     *          0.0000     0.0
0.0000    0.0       0.0000    0.0
 1.0   1      *      C2     N3     *          0.0000     0.0
0.0000    0.0       1.4000    0.0
 1.0   1      *      C2     NT     *          0.0000     0.0
0.0000    0.0       1.0000    0.0
 1.0   1      *      C2     OH     *          0.0000     0.0
0.0000    0.0       0.5000    0.0
 1.0   1      *      C2     OS     *          0.0000     0.0
0.0000    0.0       1.4500    0.0
 1.0   1      *      C2     S      *          0.0000     0.0
```

```
0.0000    0.0      1.0000    0.0
 1.0   1      *     C2     SH     *          0.0000    0.0
0.0000    0.0      0.7500    0.0
 1.0   1      *     CA     CA     *          0.0000    0.0
5.3000  180.0      0.0000    0.0
 1.0   1      *     CA     CB     *          0.0000    0.0
10.2000  180.0     0.0000    0.0
 1.0   1      *     CA     CD     *          0.0000    0.0
5.3000  180.0      0.0000    0.0
 1.0   1      *     CA     CJ     *          0.0000    0.0
3.7000  180.0      0.0000    0.0
 1.0   1      *     CA     CM     *          0.0000    0.0
3.7000  180.0      0.0000    0.0
 1.0   1      *     CA     CN     *          0.0000    0.0
10.6000  180.0     0.0000    0.0
 1.0   1      *     CA     CT     *          0.0000    0.0
0.0000    0.0      0.0000    0.0
 1.0   1      *     CA     N2     *          0.0000    0.0
6.8000  180.0      0.0000    0.0
 1.0   1      *     CA     NA     *          0.0000    0.0
6.0000  180.0      0.0000    0.0
 1.0   1      *     CA     NC     *          0.0000    0.0
9.6000  180.0      0.0000    0.0
 1.0   1      *     CB     CB     *          0.0000    0.0
16.3000  180.0     0.0000    0.0
 1.0   1      *     CB     CN     *          0.0000    0.0
20.0000  180.0     0.0000    0.0
 1.0   1      *     CB     N*     *          0.0000    0.0
6.6000  180.0      0.0000    0.0
 1.0   1      *     CB     NB     *          0.0000    0.0
5.1000  180.0      0.0000    0.0
 1.0   3      *     CB     NC     *          0.0000    0.0
8.3000  180.0      0.0000    0.0
 1.0   1      *     CC     CF     *          0.0000    0.0
14.3000  180.0     0.0000    0.0
 1.0   1      *     CC     CG     *          0.0000    0.0
15.9000  180.0     0.0000    0.0
 1.0   1      *     CC     CT     *          0.0000    0.0
0.0000    0.0      0.0000    0.0
```

```
 1.0    1       *       CC      CV      *              0.0000      0.0
14.3000   180.0        0.0000      0.0
 1.0    1       *       CC      CW      *              0.0000      0.0
15.9000   180.0        0.0000      0.0
 1.0    1       *       CC      NA      *              0.0000      0.0
5.6000   180.0        0.0000      0.0
 1.0    1       *       CC      NB      *              0.0000      0.0
4.8000   180.0        0.0000      0.0
 1.0    1       *       CD      CD      *              0.0000      0.0
5.3000   180.0        0.0000      0.0
 1.0    1       *       CD      CN      *              0.0000      0.0
5.3000   180.0        0.0000      0.0
 1.0    1       *       CE      N*      *              0.0000      0.0
6.7000   180.0        0.0000      0.0
 1.0    1       *       CE      NB      *              0.0000      0.0
20.0000   180.0        0.0000      0.0
 1.0    1       *       CF      NB      *              0.0000      0.0
4.8000   180.0        0.0000      0.0
 1.0    1       *       CG      NA      *              0.0000      0.0
6.0000   180.0        0.0000      0.0
 1.0    1       *       CH      CH      *              0.0000      0.0
0.0000     0.0        2.0000      0.0
 1.0    1       *       CH      N       *              0.0000      0.0
0.0000     0.0        0.0000      0.0
 1.0    1       *       CH      N*      *              0.0000      0.0
0.0000     0.0        0.0000      0.0
 1.0    1       *       CH      NT      *              0.0000      0.0
0.0000     0.0        1.0000      0.0
 1.0    1       *       CH      OH      *              0.0000      0.0
0.0000     0.0        0.5000      0.0
 1.0    1       *       CH      OS      *              0.0000      0.0
0.0000     0.0        1.4500      0.0
 1.0    1       *       CI      NC      *              0.0000      0.0
13.5000   180.0        0.0000      0.0
 1.0    1       *       CJ      CJ      *              0.0000      0.0
24.4000   180.0        0.0000      0.0
 1.0    1       *       CJ      CM      *              0.0000      0.0
24.4000   180.0        0.0000      0.0
 1.0    1       *       CJ      N*      *              0.0000      0.0
```

```
 7.4000  180.0      0.0000     0.0
  1.0    1      *     CK     N*     *          0.0000     0.0
 6.7000  180.0      0.0000     0.0
  1.0    1      *     CK     NB     *          0.0000     0.0
20.0000  180.0      0.0000     0.0
  1.0    1      *     CM     CM     *          0.0000     0.0
24.4000  180.0      0.0000     0.0
  1.0    1      *     CM     CT     *          0.0000     0.0
0.0000     0.0      0.0000     0.0
  1.0    1      *     CM     N*     *          0.0000     0.0
 7.4000  180.0      0.0000     0.0
  1.0    1      *     CN     NA     *          0.0000     0.0
12.2000  180.0      0.0000     0.0
  1.0    1      *     CP     NA     *          0.0000     0.0
 9.3000  180.0      0.0000     0.0
  1.0    1      *     CP     NB     *          0.0000     0.0
10.0000  180.0      0.0000     0.0
  1.0    1      *     CQ     NC     *          0.0000     0.0
13.5000  180.0      0.0000     0.0
  1.0    1      *     CR     NA     *          0.0000     0.0
 9.3000  180.0      0.0000     0.0
  1.0    1      *     CR     NB     *          0.0000     0.0
10.0000  180.0      0.0000     0.0
  1.0    1      *     CT     CT     *          0.0000     0.0
0.0000     0.0      1.3000     0.0
  1.0    1      *     CT     N      *          0.0000     0.0
0.0000     0.0      0.0000     0.0
  1.0    1      *     CT     N*     *          0.0000     0.0
0.0000     0.0      0.0000     0.0
  1.0    1      *     CT     N2     *          0.0000     0.0
0.0000     0.0      0.0000     0.0
  1.0    1      *     CT     N3     *          0.0000     0.0
0.0000     0.0      1.4000     0.0
  1.0    1      *     CT     OH     *          0.0000     0.0
0.0000     0.0      0.5000     0.0
  1.0    1      *     CT     OS     *          0.0000     0.0
0.0000     0.0      1.1500     0.0
  1.0    1      *     CT     S      *          0.0000     0.0
0.0000     0.0      1.0000     0.0
```

```
 1.0   1      *      CT     SH     *          0.0000     0.0
0.0000    0.0       0.7500    0.0
 1.0   1      *      CV     NB     *          0.0000     0.0
4.8000  180.0       0.0000    0.0
 1.0   1      *      CW     NA     *          0.0000     0.0
6.0000  180.0       0.0000    0.0
 1.0   1      *      OH     P      *          0.0000     0.0
0.0000    0.0       0.7500    0.0
 1.0   1      *      OS     P      *          0.0000     0.0
0.0000    0.0       0.7500    0.0
 1.1   4      *      CS     CS     *          0.0000     0.0
0.0000    0.0       1.0210    0.0
 1.1   4      *      CS     CT     *          0.0000     0.0
0.0000    0.0       1.0210    0.0
 1.1   4      *      AC     CS     *          0.0000     0.0
0.0000    0.0       1.0210    0.0
 1.1   4      *      BC     CS     *          0.0000     0.0
0.0000    0.0       1.0210    0.0
 1.1   4      *      CS     OT     *          0.0000     0.0
0.0000    0.0       0.4430    0.0
 1.1   4      *      CS     OE     *          0.0000     0.0
0.0000    0.0       0.9280    0.0
 1.1   4      *      AC     OE     *          0.0000     0.0
0.0000    0.0       0.9280    0.0
 1.1   4      *      BC     OE     *          0.0000     0.0
0.0000    0.0       0.9280    0.0
 1.1   4      *      AC     OA     *          0.0000     0.0
0.0000    0.0       0.0000    0.0
 1.1   4      *      BC     OB     *          0.0000     0.0
0.0000    0.0       0.0000    0.0
 1.1   4      *      CS     OA     *          0.0000     0.0
0.0000    0.0       0.0000    0.0
 1.1   4      *      CS     OB     *          0.0000     0.0
0.0000    0.0       0.0000    0.0
 1.1   4      *      CS     N      *          0.0000     0.0
0.0000    0.0       0.0000    0.0
 1.1   4      *      C      N      *          0.0000     0.0
10.0000  180.0       0.0000    0.0
 1.1   4      *      C      CS     *          0.0000     0.0
```

```
0.0000     0.0       0.0000     0.0
 1.0    1       *       CT      NT      *              0.0000      0.0
0.0000     0.0       1.8000     0.0

*************************************************************
                      DATA FILE - CX6C.CAR
*************************************************************

!BIOSYM archive 3
PBC=OFF
!DATE Thu Mar  2 10:02:29 1995
SG         0.051616628     8.775964550     2.653307337 CYSn 1
S        S    0.824
LG1       -0.116704460     8.906803991     3.732450018 CYSn 1
LP       L   -0.405
LG2       -0.816371929     8.216369655     2.274560255 CYSn 1
LP       L   -0.405
CB         1.625257994     7.970290997     2.280061368 CYSn 1
CT       C   -0.098
HB1        1.743097230     7.117856362     2.972980432 CYSn 1
HC       H    0.050
HB2        2.457560406     8.667686711     2.506611212 CYSn 1
HC       H    0.050
CA         1.664891168     7.503978115     0.811322158 CYSn 1
CT       C    0.035
HA         2.715618613     7.453348875     0.469159517 CYSn 1
HC       H    0.032
N          0.954382540     8.512673633     0.003030230 CYSn 1
NT       N   -0.463
C          1.063568189     6.132700222     0.616111991 CYSn 1
C        C    0.616
O          0.248707622     5.654726837     1.414398016 CYSn 1
O        O   -0.504
N          1.449902196     5.479885680    -0.464156147 GLY  2
N        N   -0.463
HN         2.157106102     5.992384244    -1.099457509 GLY  2
H        H    0.252
CA         0.868490592     4.154014497    -0.652902307 GLY  2
CT       C    0.035
```

```
HA1      1.550908149    3.403064022   -0.212395307 GLY  2
HC       H    0.032
HA2     -0.097660558    4.132736815   -0.116611463 GLY  2
HC       H    0.032
C        0.730531165    3.827591429   -2.120728786 GLY  2
C        C    0.616
O        1.559375145    4.206208097   -2.957020570 GLY  2
O        O   -0.504
N       -0.320742949    3.103195380   -2.456098946 GLY  3
N        N   -0.463
HN      -0.976177839    2.817016114   -1.646836012 GLY  3
H        H    0.252
CA      -0.454134161    2.787581074   -3.875321662 GLY  3
CT       C    0.035
HA1     -0.907422830    1.783240810   -3.972773051 GLY  3
HC       H    0.032
HA2     -1.127648566    3.540414569   -4.323795441 GLY  3
HC       H    0.032
C        0.896974016    2.736484179   -4.547627543 GLY  3
C        C    0.616
O        1.315189212    1.712629073   -5.101282348 GLY  3
O        O   -0.504
N        1.599575272    3.853622667   -4.520184621 GLY  4
N        N   -0.463
HN       1.137216234    4.691535216   -4.019658253 GLY  4
H        H    0.252
CA       2.905944550    3.804217731   -5.170228610 GLY  4
CT       C    0.035
HA1      3.056204584    2.789614618   -5.584558431 GLY  4
HC       H    0.032
HA2      2.897891721    4.540755026   -5.994216851 GLY  4
HC       H    0.032
C        4.014980067    4.050747291   -4.175561433 GLY  4
C        C    0.616
O        4.978871195    4.780583329   -4.436272241 GLY  4
O        O   -0.504
N        3.887759074    3.450944950   -3.006608050 GLY  5
N        N   -0.463
HN       3.003276191    2.844372268   -2.879487738 GLY  5
```

H	H	0.252			
CA		4.960071382	3.689311240	-2.044877031	GLY 5
CT	C	0.035			
HA1		5.709592998	2.881830301	-2.144167698	GLY 5
HC	H	0.032			
HA2		5.427393718	4.658369322	-2.297948016	GLY 5
HC	H	0.032			
C		4.437174470	3.643619035	-0.629041435	GLY 5
C	C	0.616			
O		3.798322352	2.676595378	-0.197242766	GLY 5
O	O	-0.504			
N		4.713663113	4.691871185	0.124033264	GLY 6
N	N	-0.463			
HN		5.286002166	5.476492875	-0.348403798	GLY 6
H	H	0.252			
CA		4.208080753	4.647691975	1.492986659	GLY 6
CT	C	0.035			
HA1		3.303800182	4.010943092	1.515218779	GLY 6
HC	H	0.032			
HA2		4.993057374	4.194323221	2.125265975	GLY 6
HC	H	0.032			
C		3.799265981	6.023038258	1.963510280	GLY 6
C	C	0.616			
O		4.006824522	7.036283245	1.285298717	GLY 6
O	O	-0.504			
N		3.195690211	6.077750863	3.136158080	GLY 7
N	N	-0.463			
HN		3.055107813	5.133307510	3.640799839	GLY 7
H	H	0.252			
CA		2.800412417	7.407555656	3.591101372	GLY 7
CT	C	0.035			
HA1		1.946687677	7.303619509	4.286815466	GLY 7
HC	H	0.032			
HA2		3.660862081	7.847316876	4.127520148	GLY 7
HC	H	0.032			
C		2.334578164	8.258959996	2.434291753	GLY 7
C	C	0.616			
O		2.337411236	9.494643783	2.487154063	GLY 7
O	O	-0.504			

```
N          1.936206121     7.605756209     1.358640986 CYSN 8
N        N  -0.463
HN         1.983632457     6.528240768     1.414418956 CYSN 8
H        H   0.252
CA         1.485796919     8.428968216     0.240136508 CYSN 8
CT       C   0.035
HA         0.399931102     8.271042216     0.100059529 CYSN 8
HC       H   0.032
C          2.167493478     8.018162291    -1.043072620 CYSN 8
C        C   0.616
CB         1.746659419     9.902481747     0.610166221 CYSN 8
CT       C  -0.098
HB1        2.709270705    10.016688002     1.140264476 CYSN 8
HC       H   0.050
HB2        1.816139488    10.541353385    -0.293951287 CYSN 8
HC       H   0.050
SG         0.440719361    10.532225816     1.688457720 CYSN 8
S        S   0.824
LG1       -0.404239097    10.957145937     1.126774557 CYSN 8
LP       L  -0.405
LG2        0.793091788    11.329491558     2.359427872 CYSN 8
LP       L  -0.405
end
```

SEQUENCE LISTING

(1) GENERAL INFORMATION:

(i) APPLICANT: Deem, Michael W.
Rothberg, Jonathan M.
Went, Gregory T.

(ii) TITLE OF INVENTION: CONSENSUS CONFIGURATIONAL BIAS MONTE CARLO METHOD AND SYSTEM FOR PHARMACOPHORE STRUCTURE DETERMINATION

(iii) NUMBER OF SEQUENCES: 10

(iv) CORRESPONDENCE ADDRESS:
(A) ADDRESSEE: Pennie & Edmonds
(B) STREET: 1155 Avenue of the Americas
(C) CITY: New York
(D) STATE: New York
(E) COUNTRY: USA
(F) ZIP: 10036-2711

(v) COMPUTER READABLE FORM:
(A) MEDIUM TYPE: Floppy disk
(B) COMPUTER: IBM PC compatible
(C) OPERATING SYSTEM: PC-DOS/MS-DOS
(D) SOFTWARE: PatentIn Release #1.0, Version #1.30

(vi) CURRENT APPLICATION DATA:
(A) APPLICATION NUMBER: To Be Assigned
(B) FILING DATE: On Even Date Herewith
(C) CLASSIFICATION:

(viii) ATTORNEY/AGENT INFORMATION:
(A) NAME: Misrock, S. Leslie
(B) REGISTRATION NUMBER: 18,872
(C) REFERENCE/DOCKET NUMBER: 7934-007

(ix) TELECOMMUNICATION INFORMATION:
(A) TELEPHONE: (212) 790-9090
(B) TELEFAX: (212) 869-9741/8864
(C) TELEX: 66141 PENNIE

(2) INFORMATION FOR SEQ ID NO:1:

(i) SEQUENCE CHARACTERISTICS:
(A) LENGTH: 8 amino acids
(B) TYPE: amino acid
(D) TOPOLOGY: unknown

(ii) MOLECULE TYPE: peptide

(ix) FEATURE:
(A) NAME/KEY: Disulfide-bond
(B) LOCATION: 1..8
(D) OTHER INFORMATION: /note= "A disulfide bond is formed between the cysteine residues."

(xi) SEQUENCE DESCRIPTION: SEQ ID NO:1:

Cys Xaa Xaa Xaa Xaa Xaa Xaa Cys
1 5

(2) INFORMATION FOR SEQ ID NO:2:

(i) SEQUENCE CHARACTERISTICS:
(A) LENGTH: 102 base pairs
(B) TYPE: nucleic acid
(C) STRANDEDNESS: single
(D) TOPOLOGY: linear

(ii) MOLECULE TYPE: DNA

(xi) SEQUENCE DESCRIPTION: SEQ ID NO:2:

ACTTCGAAAT TAATACGACT CACTATAGGG AGACCACAAC GGTTTCCCTC CAGAAATAAT 60

TTTGTTTAAC TTTAACTTTA AGAAGGAGAT ATACATATGC AT 102

(2) INFORMATION FOR SEQ ID NO:3:

(i) SEQUENCE CHARACTERISTICS:
(A) LENGTH: 83 base pairs
(B) TYPE: nucleic acid
(C) STRANDEDNESS: single
(D) TOPOLOGY: linear

(ii) MOLECULE TYPE: DNA

(xi) SEQUENCE DESCRIPTION: SEQ ID NO:3:

CCCAGACCCG CCCCCAGCAT TGTGGGTTCC AACGCCCTCT AGACAMNNMN NMNNMNNMNN 60

MNNACAATGT ATATCTCCTT CTT 83

(2) INFORMATION FOR SEQ ID NO:4:

(i) SEQUENCE CHARACTERISTICS:
(A) LENGTH: 48 base pairs
(B) TYPE: nucleic acid
(C) STRANDEDNESS: single
(D) TOPOLOGY: linear

(ii) MOLECULE TYPE: DNA

(xi) SEQUENCE DESCRIPTION: SEQ ID NO:4:

TCGTCTGACC TGCCTCAACC TCCCCACAAT GCTGGCGGCG GCTCTGGT 48

(2) INFORMATION FOR SEQ ID NO:5:

(i) SEQUENCE CHARACTERISTICS:
(A) LENGTH: 42 base pairs
(B) TYPE: nucleic acid
(C) STRANDEDNESS: single
(D) TOPOLOGY: linear

(ii) MOLECULE TYPE: DNA

(xi) SEQUENCE DESCRIPTION: SEQ ID NO:5:

ATCAAGTTTG CCTTTACCAG CATTGTGGAG CGCGTTTTCA TC 42

(2) INFORMATION FOR SEQ ID NO:6:

(i) SEQUENCE CHARACTERISTICS:
(A) LENGTH: 10 amino acids
(B) TYPE: amino acid
(D) TOPOLOGY: unknown

(ii) MOLECULE TYPE: peptide

(xi) SEQUENCE DESCRIPTION: SEQ ID NO:6:

Met His Cys Xaa Xaa Xaa Xaa Xaa Xaa Cys
1 5 10

(2) INFORMATION FOR SEQ ID NO:7:

(i) SEQUENCE CHARACTERISTICS:
(A) LENGTH: 8 amino acids
(B) TYPE: amino acid
(D) TOPOLOGY: unknown

(ii) MOLECULE TYPE: peptide

(xi) SEQUENCE DESCRIPTION: SEQ ID NO:7:

Cys Gly Gly Gly Gly Gly Gly Cys
1 5

(2) INFORMATION FOR SEQ ID NO:8:

(i) SEQUENCE CHARACTERISTICS:
(A) LENGTH: 30 base pairs
(B) TYPE: nucleic acid
(C) STRANDEDNESS: single
(D) TOPOLOGY: unknown

(ii) MOLECULE TYPE: DNA

(xi) SEQUENCE DESCRIPTION: SEQ ID NO:8:

NNKNNKNNKN NKNNKNNKNN KNNKNNKNNK 30

(2) INFORMATION FOR SEQ ID NO:9:

(i) SEQUENCE CHARACTERISTICS:
(A) LENGTH: 47 base pairs
(B) TYPE: nucleic acid
(C) STRANDEDNESS: single
(D) TOPOLOGY: linear

(ii) MOLECULE TYPE: DNA

(xi) SEQUENCE DESCRIPTION: SEQ ID NO:9:

ACTTCGAAAT TAATACGACT CACTATAGGG AGACCACAAC GGTTTCC 47

(2) INFORMATION FOR SEQ ID NO:10:

(i) SEQUENCE CHARACTERISTICS:
(A) LENGTH: 9 amino acids
(B) TYPE: amino acid
(D) TOPOLOGY: unknown

(ii) MOLECULE TYPE: peptide

(xi) SEQUENCE DESCRIPTION: SEQ ID NO:10:

Cys Asn Thr Leu Lys Gly Asp Cys Gly
1 5

WHAT IS CLAIMED IS:

1. A method of determining a consensus pharmacophore structure comprising the steps of:
   (a) identifying from one or more diversity libraries a plurality of compounds that bind to a target molecule,
   (b) measuring one or more distances in one or more of the compounds, and
   (c) determining a consensus pharmacophore structure for the compounds.

2. The method of claim 1 wherein said compounds are peptides, peptide derivatives, or peptide analogs.

3. The method of claim 2 wherein said compounds are peptides containing one or more cystines.

4. The method of claim 3 wherein the peptides comprise the sequence CX_6C (SEQ ID NO:1).

5. The method of claim 1 further comprising a step of selecting a plurality of candidate pharmacophores based on rules of chemical homology, the selected plurality of candidate pharmacophores being used in step (c) to determine the consensus pharmacophore structure.

6. The method of claim 5 wherein the rules of homology determine that two candidate pharmacophores are homologous if they have chemically similar side chains.

7. The method of claim 1 which further comprises after said identifying step, a screening step involving a genetic selection technique.

8. The method of claim 1 wherein the step of measuring distance comprises making solid phase nuclear magnetic

resonance measurements on selected nuclei in a nuclear magnetic resonance spectrometer upon a sample comprising one of the compounds.

9. The method of claim 8 wherein the step of measuring distances further comprises making rotational echo double resonance nuclear magnetic resonance measurements of internuclear dipole-dipole interaction strength between selected nuclei in the compound in the sample.

10. The method of claim 8 wherein the sample further comprises a substrate having a surface to which the compound is attached.

11. The method of claim 8 wherein the sample is cooled below room temperature.

12. The method of claim 8 wherein the compound is bound to the target molecule.

13. The method of claim 10 wherein a plurality of the compound is attached to the surface at a surface density such that the inter-nuclear dipole-dipole interactions between different molecules is less than 10% of the inter-nuclear dipole-dipole interaction within one molecule.

14. The method of claim 10 wherein the substrate has pores of sufficient size to permit the target to diffuse and bind to the compound in the sample.

15. The method of claim 9 wherein rotational echo double resonance nuclear magnetic resonance measurements can be made on the compound bound to the target or hydrated or in a dry nitrogen atmosphere.

16. The method of claim 10 wherein the compound is a peptide, and a plurality of the peptide is attached to the substrate surface, which has a purity of the peptide of at least 95% and wherein the surface density of the peptide is no more than one peptide per 100 $Å^2$ of substrate surface.

17. The method of claim 10 wherein the substrate is selected from the group consisting of p-MethylBenzhydrilamine resin, divinylbenzyl polystyrene resin, and glass beads.

18. The method of claim 8 wherein the selected nuclei are selected from the group consisting of ^{13}C, ^{15}N, ^{19}F, and ^{31}P.

19. The method of claim 9 wherein the nuclear magnetic resonance spectrometer comprises magnetic excitation means, a sample rotor, and free induction decay observing means, and the step of making rotational echo double resonance nuclear magnetic resonance measurements further comprises the steps of:

(a) spinning the sample in the sample rotor,

(b) initially exciting magnetically the selected nuclei to be observed,

(c) providing subsequently one π spin flip magnetic excitation during each rotor period to each of the selected nuclei, the pulses to the different nuclei having fixed phase delays,

(d) observing the free induction decay signal as a function of the number of rotor periods; and

(e) finding the dipole-dipole strength between the selected nuclei, whereby the internuclear distance between the selected nuclei can be obtained.

20. The method of claim 1 wherein the step of measuring distances comprises making liquid phase nuclear magnetic resonance measurements.

21. A method of determining a consensus pharmacophore structure comprising the steps of:
(a) identifying from one or more diversity libraries a plurality of compounds that bind to a target molecule,
(b) determining a consensus pharmacophore structure for the compounds.

22. A method of determining a consensus pharmacophore structure comprising the steps of:
(a) measuring one or more distances in one or more compounds that bind to a target molecule, and
(b) determining a consensus pharmacophore structure for the compounds.

23. The method of claim 21 or 22 further comprising a step of selecting a plurality of candidate pharmacophores based on rules of chemical homology, the selected plurality of candidate pharmacophores being used in step (b) to determine the consensus pharmacophore structure.

24. The method of claim 23 wherein the compounds have limited conformational degrees of freedom at the temperature of interest, and wherein the step of determining a consensus pharmacophore structure for each compound further comprises, performing a consensus configurational bias Monte Carlo method, said Monte Carlo method comprising the steps of:
(a) generating a proposed structure for a compound identified from said one or more diversity libraries by making conformational alterations consistent with the conformational degrees of freedom, the alterations being made to a representation of the compound's current chemical and conformational structure to generate a proposed representation, the proposed structure being generated with a bias

toward more acceptable configurations of lower energy, whereby the method is made more efficient,

(b) accepting and storing the proposed structure according to a probability depending on an energy determined for the proposed structure, and

(c) repeating these steps until sufficient structures have been stored for each compound to permit statistically significant determination of an equilibrium structure for each compound.

25. A method of determining one or more lead compounds for use as a drug that binds to a target molecule comprising the steps of:

(a) identifying from one or more diversity libraries a plurality of compounds that bind to a target molecule;

(b) determining a consensus pharmacophore structure for the compounds; and

(c) determining one or more lead compounds for use as a drug which share a pharmacophore specification with the determined consensus pharmacophore structure.

26. A method of determining one or more lead compounds for use as a drug that binds to a target molecule comprising the steps of:

(a) measuring one or more distances in one or more compounds that bind to a target molecule;

(b) determining a consensus pharmacophore structure for the compounds; and

(c) determining one or more lead compounds for use as a drug which share a pharmacophore specification with the determined consensus pharmacophore structure.

27. The method according to claim 25 or 26 wherein said step of determining one or more lead compounds comprises modifying a compound identified as binding to the target molecule, said modification being done outside of the

pharmacophore structure, to render the compound more attractive for use as a drug.

28. The method of claim 5 wherein the compounds have limited conformational degrees of freedom at a temperature of interest, and wherein the step of determining a consensus pharmacophore structure for the compounds further comprises performing a consensus configurational bias Monte Carlo method, said Monte Carlo method comprising the steps of:

(a) generating a proposed structure for a compound identified from said one or more diversity libraries by making conformational alterations consistent with the conformational degrees of freedom, the alterations being made to a representation of the compound's current chemical and conformational structure to generate a proposed representation, the proposed structure being generated with a bias toward more acceptable configurations of lower energy,

(b) accepting and storing the proposed structure according to a probability depending on an energy determined for the proposed structure, and

(c) repeating these steps until sufficient structures have been stored for each compound to permit statistically significant determination of an equilibrium structure for each compound.

29. The method of claim 28 wherein the limited conformational degrees of freedom comprise torsional rotations about mutual bonds between otherwise rigid subunits of the compound, each rigid unit's representation comprising its interconnections and atomic composition, each atom's representation comprising its type and position, the torsional rotations respecting any conformational constraints present.

30. The method of claim 28 wherein the compound is a peptide, peptide derivative, or peptide analog.

31. The method of claim 28 wherein the conformational alterations comprise constrained, concerted torsional rotations or removal of a side chain and regrowth of the side chain with a new torsional conformation.

32. The method of claim 31 wherein the constrained, concerted torsional rotations are constrained so that no more than four rigid units are spatially displaced.

33. The method of claim 28 wherein determining the energy for the proposed structure of one compound comprises including one or more constraint terms which represent knowledge of measured structure for the compound.

34. The method of claim 33 wherein the constraint terms comprise a weighted sum of squares of differences of the actual and measured structures.

35. The method of claim 28 wherein the energy is determined for the proposed structure of one compound by a method comprising including consensus terms which represent knowledge that the identified compounds all bind to the same target, the compounds being otherwise treated independently by the method.

36. The method of claim 35 wherein the consensus terms are a weighted sum of squares of differences in the atomic positions of a candidate pharmacophore from the average values of these positions in all the compounds.

37. The method of claim 35 wherein the step of determining the consensus pharmacophore structure comprises determining from the plurality of selected candidate pharmacophores a candidate pharmacophore for which the

consensus terms are relatively small compared to the total energy.

38. The method of claim 35 wherein the step of determining the consensus pharmacophore structure comprises determining from the plurality of selected candidate pharmacophores a candidate pharmacophore for which the consensus terms are minimum compared to other selected regions.

39. The method of claim 28 wherein the equilibrium structure is determined by a method comprising averaging selected generated and accepted structures for each compound.

40. The method of claim 39 wherein the averaging of structures comprises clustering selected generated and accepted structures into sets of similar structures and averaging these sets for each member.

41. A method of identifying a compound that binds to a target molecule comprising the following steps in the order stated:

(a) contacting compounds of a phage display or polysome-based diversity library with a target molecule;

(b) identifying one or more compounds in the library that bind to the target molecule;

(c) contacting one or more first fusion proteins, each first fusion protein comprising an identified compound, with a second fusion protein comprising the target molecule or a binding portion thereof, in which binding of the first fusion protein to the second fusion protein results in an increase in activity or activation of a transcriptional promoter or an origin of replication; and

(d) identifying one or more of the compounds that when present in said first fusion protein result in said increase in activity or activation.

42. A method of making solid state nuclear magnetic resonance measurements comprising measuring internuclear dipole-dipole interaction strengths between selected nuclei in a compound, said compound being attached to the surface of a substrate.

43. The method of claim 42 which further comprises before said measuring step the step of synthesizing a plurality of said compound on the surface of the substrate.

44. The method of claim 43 wherein said plurality of the compound is at least 95% pure.

45. The method of claim 42 wherein a plurality of said compound is attached to the substrate surface, with at least 10 Å spacing between molecules of the compound.

46. The method of claim 42 wherein the substrate has pores of sufficient size to permit a molecule to diffuse and bind to the compound.

47. The method of claim 42 wherein the substrate has a surface density of the compound such that the internuclear dipole-dipole interactions between different molecules of the compound is less than 10% of the internuclear dipole-dipole interaction within one molecule of the compound.

48. The method of claim 42 wherein the compound is a peptide, peptide derivative, or peptide analog.

49. The method of claim 42 wherein the substrate is selected from the group consisting of p-MethylBenzhydrilamine resin, divinylbenzyl polystyrene resin, and a glass bead.

50. The method of claim 42 wherein said measuring step comprises using a nuclear magnetic resonance

spectrometer, said spectrometer comprising magnetic excitation means, a sample rotor, and free induction decay observing means; and said measurement of internuclear dipole-dipole interaction is done by a method comprising the steps of:

(a) spinning the sample in the sample rotor;

(b) initially exciting magnetically the selected nuclei to be observed;

(c) providing subsequently one or more π spin flip magnetic excitations during each rotor period to one or both of the selected nuclei, wherein pulses to the different nuclei have fixed phase delays;

(d) observing a free induction decay signal as a function of the number of rotor periods; and

(e) determining the dipole-dipole strength between the selected nuclei, whereby the internuclear distance between the selected nuclei can be obtained.

51. A method of configurational bias Monte Carlo determination of the structure of a compound having limited conformational degrees of freedom at a temperature of interest, the method comprising the steps of:

(a) generating a proposed structure for the compound by making conformational alterations consistent with the conformational degrees of freedom, the alterations being made to a representation of the compound's current chemical and conformational structure to generate a proposed representation;

(b) accepting and storing the proposed structure according to a probability depending on an energy determined for the proposed structure; and

(c) repeating these steps until sufficient structures have been stored to permit statistically significant determination of an equilibrium structure.

52. The method of claim 51 wherein the conformational degrees of freedom comprise torsional rotations about mutual bonds between otherwise rigid subunits of the compound, each rigid unit's representation comprising its interconnections and atomic composition, each atom's representation comprising its type and position, the torsional rotations respecting any conformational constraints present.

53. The method of claim 51 wherein the compound is a peptide, peptide derivative, or peptide analog.

54. The method of claim 51 wherein the conformational alterations comprise constrained, concerted torsional rotations.

55. The method of claim 54 wherein the constrained, concerted torsional rotations are constrained so that no more than four rigid units are spatially displaced.

56. The method of claim 51 wherein the conformational alterations comprise removal of a side chain and regrowth of the side chain with a new torsional conformation.

57. The method of claim 51 wherein the proposed structures are generated with a bias toward more acceptable configurations of lower energy.

58. The method of claim 51 wherein the energy is determined for the proposed structure by a method comprising including constraint terms which represent knowledge of measured structure for the compound.

59. The method of claim 58 wherein the constraint terms comprise a weighted sum of squares of differences of the actual and measured structures.

60. The method of claim 51 applied to a plurality of compounds of limited conformational degrees of freedom all of which bind to the same target molecule wherein the method further comprises a step of selecting a plurality of candidate pharmacophores based on rules of chemical homology.

61. The method of claim 60 wherein the energy is determined for the proposed structure of one of the plurality of compounds by a method comprising including consensus terms which represent knowledge that the compounds all bind to the same target molecule.

62. The method of claim 61 wherein the consensus terms are a weighted sum of squares of differences in the atomic positions of a candidate pharmacophore of said one of the plurality of compounds from the average values of these positions in all the compounds.

63. The method of claim 61 which further comprises a step of determining a consensus pharmacophore structure by determining from the plurality of selected candidate pharmacophores that candidate pharmacophore for which the consensus terms are minimum compared to other candidate pharmacophores.

64. The method of claim 60 which further comprises a step of determining a consensus pharmacophore structure by determining from the plurality of selected candidate pharmacophores that candidate pharmacophore for which the consensus terms are relatively small compared to the total energy.

65. The method of claim 63 or 64 which further comprises a step of determining one or more lead compounds for use as a drug which share a pharmacophore specification with the determined consensus pharmacophore structure.

66. The method of claim 51 wherein the equilibrium structure is determined by a method comprising averaging selected generated and accepted structures.

67. The method of claim 66 wherein the averaging of structures comprises clustering selected generated and accepted structures into sets of similar structures and averaging these sets.

68. An apparatus for configurational bias Monte Carlo determination of the structure of a compound having limited conformational degrees of freedom at a temperature of interest, the apparatus comprising:

(a) memory means for storing

(i) data structures representing the compound's chemical and conformational structure consistently with the compound's degrees of freedom,

(ii) similar data structures representing the compound's proposed structure and prior structures, and

(iii) parameters representing atomic interactions, and

(b) processor means for executing programs for

(i) generating a proposed structure by making conformational alterations consistent with the conformational degrees of freedom and with a bias toward more acceptable configurations of lower energy,

(ii) accepting and storing the proposed structure according to a probability depending on an energy determined for the proposed structure, and

(iii) repeating these steps until sufficient structures have been stored to permit statistically significant determination of an equilibrium structure.

69. The apparatus of claim 68 wherein the conformational degrees of freedom comprise torsional rotations about mutual bonds between otherwise rigid subunits of the compound, each rigid unit's representation comprising its interconnections and atomic composition, each atom's representation comprising its type and position, the torsional rotations respecting any conformational constraints present.

70. The apparatus of claim 68 wherein the compound is a peptide, peptide derivative, or peptide analog.

71. The apparatus of claim 68 wherein the memory, processor, and control means are configured from a workstation type digital computer comprising RAM memory, disk memory, processor, and input and display devices.

72. The apparatus of claim 68 wherein the conformational alterations made by the processor means further comprise constrained, concerted torsional rotations or removal of a side chain and regrowth of the side chain with a new torsional conformation.

73. The apparatus of claim 72 wherein the constrained, concerted torsional rotations are constrained so that no more than four rigid units are spatially displaced.

74. The apparatus of claim 68 wherein the processor means determines an energy for the proposed structure by a method comprising including constraint terms which represent knowledge of measured structure for the compound.

75. The apparatus of claim 74 wherein the constraint terms comprise a weighted sum of squares of differences of the actual and measured structures.

76. The apparatus of claim 68 applied to a plurality of compounds of limited conformational degrees of freedom all of which bind to the same target molecule, and wherein the processor means further comprises programs for selecting a plurality of candidate pharmacophores based on rules of chemical homology.

77. The apparatus of claim 76 wherein the processor means determines an energy for the proposed structure of any one compound by a method comprising including consensus terms which represent knowledge that the compounds all bind to the same target molecule.

78. The apparatus of claim 77 wherein the consensus terms are a weighted sum of squares of differences in the atomic positions of the candidate pharmacophore of said one compound from the average values of these positions in all the compounds.

79. The apparatus of claim 77 wherein the processor means further comprises programs for determining a consensus pharmacophore structure by determining from the plurality of selected candidate pharmacophores a candidate pharmacophore for which the consensus terms are minimum compared to other candidate pharmacophores.

80. The apparatus of claim 77 wherein the processor means further comprises programs for determining a consensus pharmacophore structure by determining from the plurality of selected candidate pharmacophores a candidate pharmacophore for which the consensus terms are relatively small compared to the total energy.

81. The apparatus of claim 79 or 80 wherein the processor means further comprises programs for determining one or more lead compounds for use as a drug that share a

pharmacophore specification with the consensus pharmacophore structure.

82. The apparatus of claim 68 wherein the processor means determines an equilibrium structure by a method comprising averaging selected generated and accepted structures.

83. The apparatus of claim 82 wherein the averaging of structures further comprises clustering selected generated and accepted structures into sets of similar structures and averaging these sets.

84. In a digital computer, apparatus for configurational bias Monte Carlo determination of the structure of at least one compound having limited conformational degrees of freedom at a temperature of interest, said apparatus comprising:

(a) first memory means for storing data structures representing the compound's chemical and conformational structure consistently with the compound's degrees of freedom,

(b) second memory means for storing similar data structures representing the compound's proposed structure,

(c) third memory means for storing similar data structures representing the compound's prior structures,

(d) first processor means for generating a proposed structure by making conformational alterations consistent with the conformational degrees of freedom and with a bias toward conformations of lower energy,

(e) second processor means for accepting and storing the proposed structure according to a probability depending on an energy determined for the proposed structure, and

(f) third processor means for controlling and repeating the generation and acceptance until sufficient structures have been stored to permit statistically significant determination of an equilibrium structure.

85. The digital computer apparatus of claim 84 wherein the conformational degrees of freedom comprise torsional rotations about mutual bonds between otherwise rigid subunits of the compound, each rigid unit's representation comprising its interconnections and atomic composition, each atom's representation comprising its type and position, the torsional rotations respecting any conformational constraints present.

86. The digital computer apparatus of claim 84 wherein the compound is a peptide, peptide derivative, or peptide analog.

87. The digital computer apparatus of claim 84 wherein the digital computer is a workstation type digital computer comprising RAM memory, disk memory, processor, and input and display devices.

88. The digital computer apparatus of claim 84 wherein the conformational alterations generated by the first processor means comprise constrained, concerted torsional rotations or removal of a side chain and regrowth of the side chain with a new torsional conformation.

89. The digital computer apparatus of claim 88 wherein the constrained, concerted torsional rotations are constrained so that no more than four rigid units are spatially displaced.

90. The digital computer apparatus of claim 84 wherein the second processor means determines an energy for the

proposed structure by a method comprising including constraint terms which represent knowledge of measured structure for the compound.

91. The digital computer apparatus of claim 90 wherein the constraint terms comprise a weighted sum of squares of differences of the actual and measured structures.

92. The digital computer apparatus of claim 84 in which said at least one compound is a plurality of compounds of limited conformational degrees of freedom all of which bind to the same target and wherein data are stored in said first memory means representing the chemical and conformational structure of said plurality of compounds and wherein the apparatus further comprises additional processor means for selecting a plurality of candidate pharmacophores based on rules of chemical homology.

93. The digital computer apparatus of claim 92 wherein the second processor means determines an energy for the proposed structure of one of said plurality of compounds by a method comprising including consensus terms which represent knowledge that the compounds all bind to the same target molecule.

94. The digital computer apparatus of claim 92 wherein the consensus terms are a weighted sum of squares of differences in the atomic positions of a candidate pharmacophore of said one of the plurality of compounds from the average values of these positions in all the compounds.

95. The digital computer apparatus of claim 93 wherein the apparatus further comprises processor means for determining a consensus pharmacophore structure by determining from the plurality of selected candidate pharmacophores a candidate pharmacophore for which the

consensus terms are relatively small compared to the total energy.

96. The digital computer apparatus of claim 93 wherein the apparatus further comprises processor means for determining a consensus pharmacophore structure by determining from the plurality of selected candidate pharmacophores a candidate pharmacophore for which the consensus terms are minimum compared to other candidate pharmacophores.

97. The digital computer apparatus of claims 95 or 96 wherein the apparatus further comprises processor means for determining one or more lead compounds for use as a drug that share a pharmacophore specification with the consensus pharmacophore structure.

98. The digital computer apparatus of claim 84 wherein the third processor means determines an equilibrium structure by a method comprising averaging selected generated and accepted structures.

99. The digital computer apparatus of claim 98 wherein the averaging of structures comprises clustering selected generated and accepted structures into sets of similar structures and averaging these sets.

100. In a digital computer, apparatus for configurational bias Monte Carlo determination of the structure of a plurality of compounds having limited conformational degrees of freedom, each compound having a backbone and side chains, said apparatus comprising:

(a) first memory means for storing data structures representing each compound's chemical and conformational structure consistently with that compound's degrees of freedom and constraints,

(b) second memory means for storing similar data structures representing a proposed structure for one or more of the compounds,

(c) third memory means for storing similar data structures representing prior structures of the plurality of compounds,

(d) first processor means for generating a proposed structure of a randomly selected compound by making conformational alterations consistent with the conformational degrees of freedom, the conformational alterations being randomly distributed between alterations that alter the structure of a randomly selected side chain of the selected compound and alterations that alter the structure of a randomly selected region of the backbone of the selected compound, the proposed structure being stored in the second memory means, the proposed structure being generated with a bias toward more acceptable structures of lower energy, whereby the method is made more efficient,

(e) second processor means for accepting a proposed structure according to a probability depending on an energy determined for the proposed structure, the energy including terms representing physical interactions and terms representing heuristic information about the compound's structure, the heuristic information comprising knowledge about measured distances in one or more compounds of said plurality and about the plurality of the compounds binding to a same target molecule,

(f) third processor means for controlling and repeating these steps until sufficient structures have been generated and accepted to permit statistically significant determination of an equilibrium structure.

101. The digital computer of claim 100 wherein the conformational degrees of freedom comprise torsional rotations about mutual bonds between otherwise rigid subunits of the compound, each rigid unit's representation comprising its interconnections and atomic composition, each atom's representation comprising its type and position, the torsional rotations respecting any conformational constraints present.

102. The digital computer of claim 100 wherein the compound is a peptide, peptide derivative, or peptide analog.

103. A method of configurational bias Monte Carlo determination of the structure of a compound selected from the group consisting of a peptide, peptide derivative, and peptide analog, the method comprising the steps of:

(a) representing the conformation of the compound by interconnected rigid units capable of torsional rotation about common bonds, each rigid unit's representation comprising its interconnections and atomic composition, each atom's representation comprising its type and position,

(b) generating a proposed structure by making conformational alterations consistent with the compound's structure,

(c) accepting a proposed structure according to a probability depending on an energy determined for the proposed structure, and

(d) repeating these steps until sufficient structures have been generated and accepted to permit statistically significant determination of an equilibrium structure.

104. An apparatus for configurational bias Monte Carlo determination of the structure of a compound selected

from the group consisting of a peptide, peptide derivative, and peptide analog, the apparatus comprising:

(a) memory means for storing

(i) data structures representing the compound's conformation as interconnected rigid units capable of torsional rotation about common bonds, each rigid unit's representation comprising its interconnections and atomic composition, each atom's representation comprising its type and position,

(ii) similar data structures representing the compound's proposed structure and prior structures, and

(iii) parameters representing atomic interactions, and

(b) processor means for executing programs for

(i) generating a proposed structure by making conformational alterations consistent with the compound's structure and with a bias toward more acceptable configurations of lower energy,

(ii) accepting a proposed structure according to a probability depending on an energy determined for the proposed structure, and

(iii) repeating these steps until sufficient structures have been generated and accepted to permit statistically significant determination of an equilibrium structure.

105. In a digital computer, apparatus for configurational bias Monte Carlo determination of the structure of a compound selected from the group consisting of a peptide, peptide derivative, and peptide analog, said apparatus comprising:

(a) first memory means for storing data structures representing the compound's structure as interconnected rigid units capable of torsional rotation about common bonds, each rigid unit's

representation comprising its interconnections and atomic composition, each atom's representation comprising its type and position,

(b) second memory means for storing similar data structures representing the compound's proposed structure,

(c) third memory means for storing similar data structures representing the compound's prior structures,

(d) first processor means for generating a proposed structure by making conformational alterations consistent with the compound's structure and constraints and with a bias toward conformations of lower energy,

(e) second processor means for accepting a proposed structure according to a probability depending on an energy determined for the proposed structure, and

(f) third processor means for controlling and repeating these steps until sufficient structures have been generated and accepted to permit statistically significant determination of an equilibrium structure.

106. In a digital computer, apparatus for configurational bias Monte Carlo determination of the structure of a plurality of compounds selected from the group consisting of peptides, peptide derivatives, and peptide analogs, each compound having a backbone and side chains, said apparatus comprising:

(a) first memory means for storing data structures representing each compound's structure as interconnected rigid units capable of torsional rotation about common bonds, each rigid unit's representation comprising its interconnections and atomic composition, each atom's representation comprising its type and position,

(b) second memory means for storing similar data structures representing a proposed structure for one or more of the compounds,

(c) third memory means for storing similar data structures representing prior structures of the plurality of the compounds,

(d) first processor means for generating a proposed structure of a randomly selected compound by making conformational alterations consistent with the compound's structure, the conformational alterations being randomly distributed between alterations that alter the structure of a randomly selected side chain of the selected compound and alterations that alter the structure of a randomly selected region of the backbone of the selected compound, the proposed structure being stored in the second memory means, the proposed structure being generated with a bias toward more acceptable structures of lower energy,

(e) second processor means for accepting a proposed structure according to a probability depending on an energy determined for the proposed structure, the energy including terms representing physical interactions and terms representing heuristic information about the compound's structure, the heuristic information comprising knowledge about measured distances in one or more compounds of said plurality and about the plurality of the compounds binding to a same target molecule,

(f) third processor means for controlling and repeating these steps until sufficient structures have been generated and accepted to permit statistically significant determination of an equilibrium structure.

107. The method of claim 42 wherein the nuclear magnetic resonance is rotational echo double resonance.

108. The method of claim 1 wherein the diversity libraries are structurally constrained organic diversity libraries.

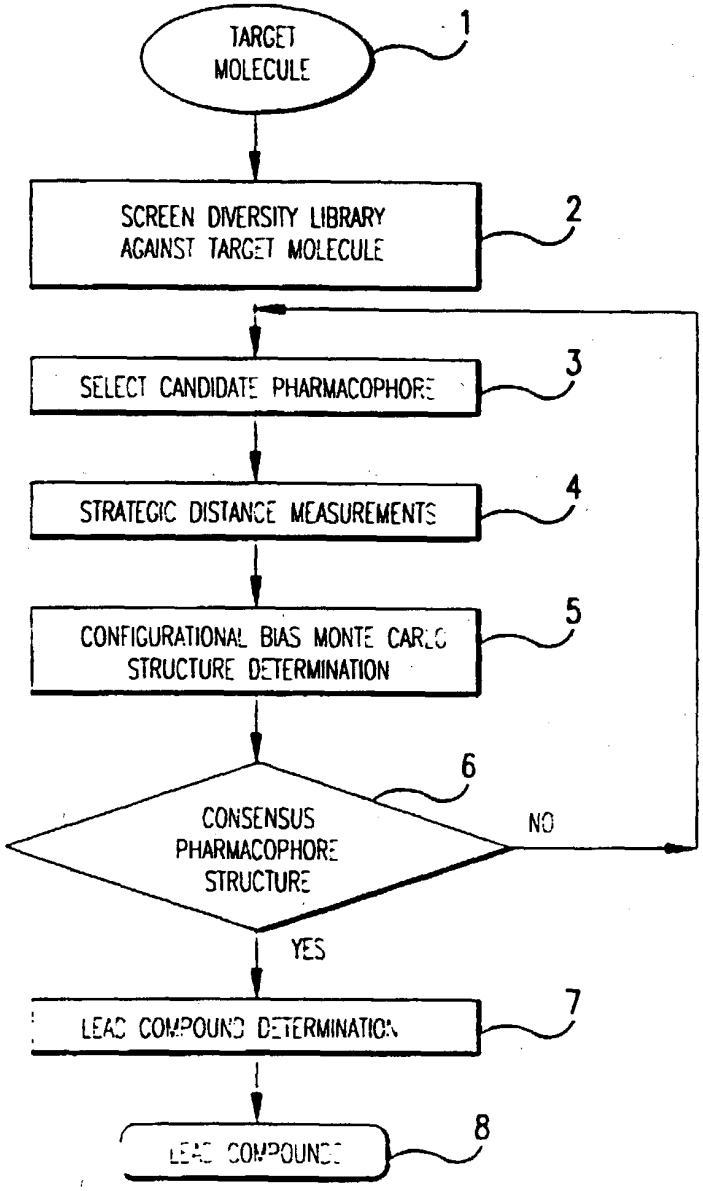
TARGET MOLECULE
1
SCREEN DIVERSITY LIBRARY AGAINST TARGET MOLECULE
2
SELECT CANDIDATE PHARMACOPHORE
3
STRATEGIC DISTANCE MEASUREMENTS
4
CONFIGURATIONAL BIAS MONTE CARLO STRUCTURE DETERMINATION
5
CONSENSUS PHARMACOPHORE STRUCTURE
6
NO
YES
LEAD COMPOUND DETERMINATION
7
LEAD COMPOUNDS
8

FIG.1

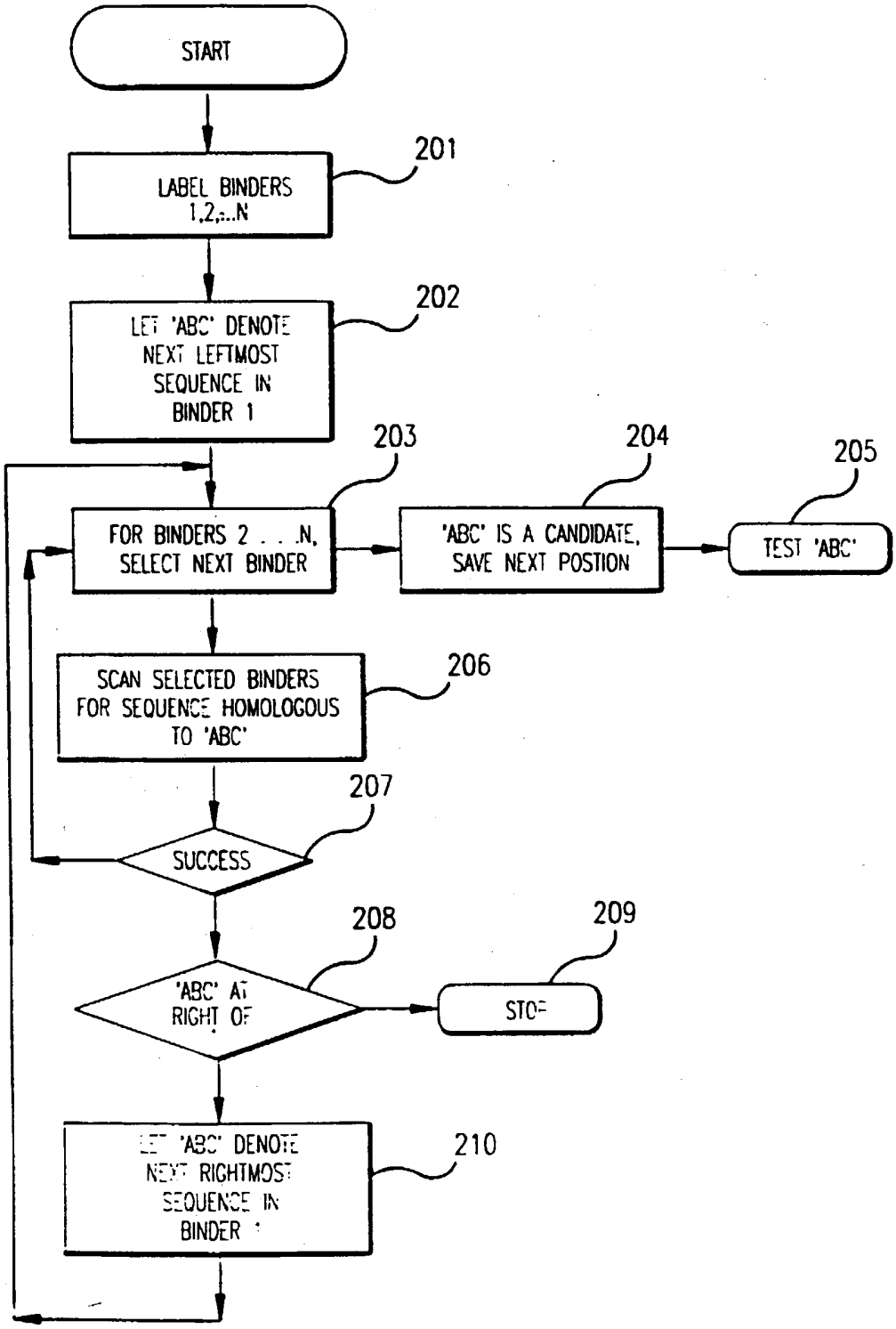
START
LABEL BINDERS 1,2,...N
201
LET 'ABC' DENOTE NEXT LEFTMOST SEQUENCE IN BINDER 1
202
FOR BINDERS 2 . . .N, SELECT NEXT BINDER
203
'ABC' IS A CANDIDATE, SAVE NEXT POSTION
204
TEST 'ABC'
205
SCAN SELECTED BINDERS FOR SEQUENCE HOMOLOGOUS TO 'ABC'
206
SUCCESS
207
'ABC' AT RIGHT OF
208
STOP
209
LET 'ABC' DENOTE NEXT RIGHTMOST SEQUENCE IN BINDER
210

FIG.2A

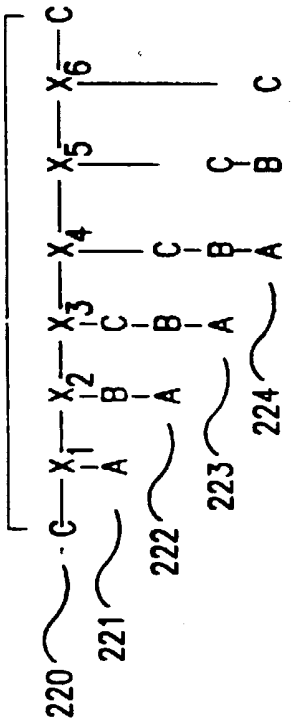
C X1 X2 X3 X4 X5 X6 C
A B C A B C C
B C A B C A B
A B C A B C C
220
221
222
223
224

FIG.2B

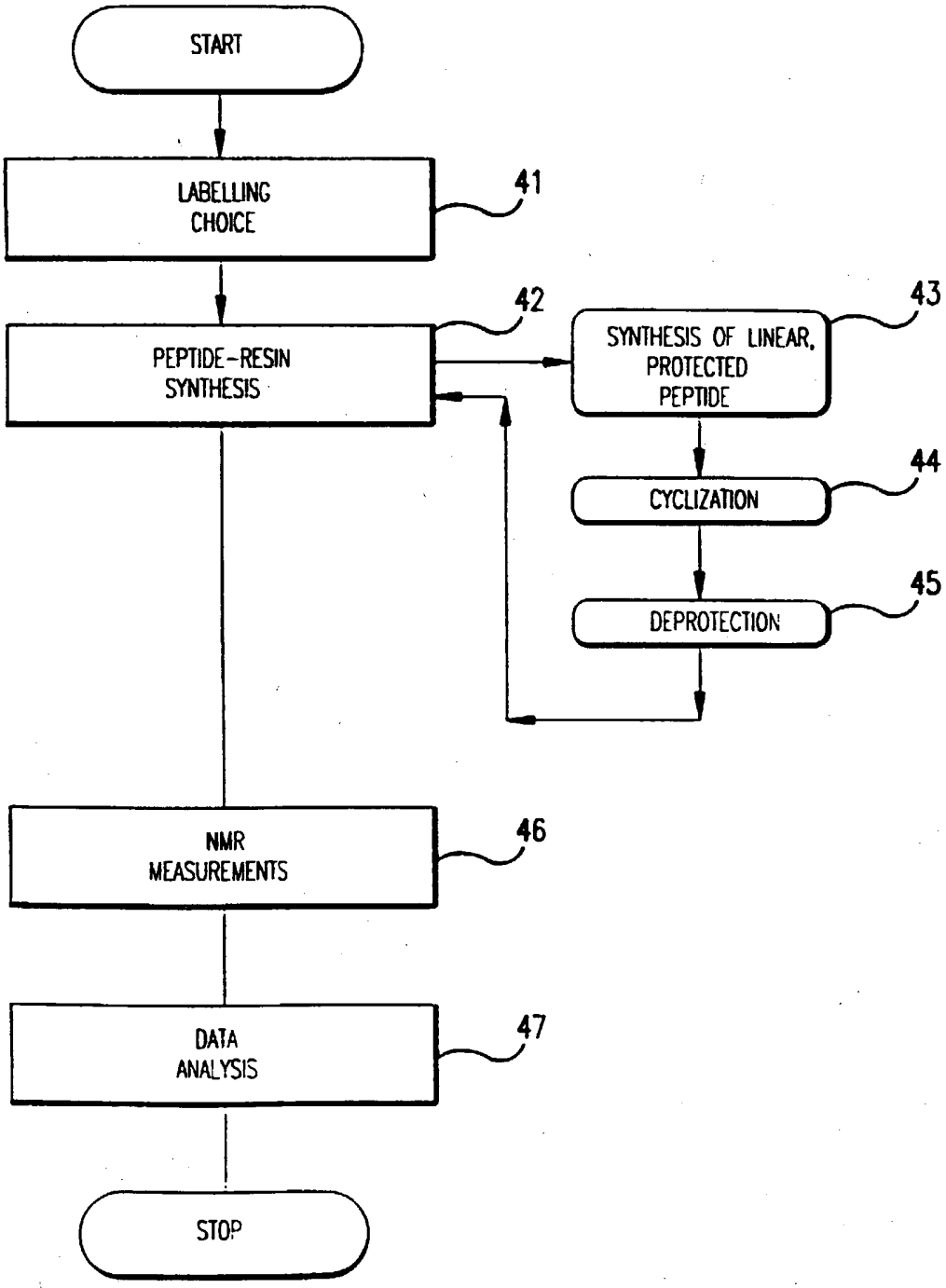
START
LABELLING CHOICE
41
PEPTIDE-RESIN SYNTHESIS
42
SYNTHESIS OF LINEAR, PROTECTED PEPTIDE
43
CYCLIZATION
44
DEPROTECTION
45
NMR MEASUREMENTS
46
DATA ANALYSIS
47
STOP

FIG.3

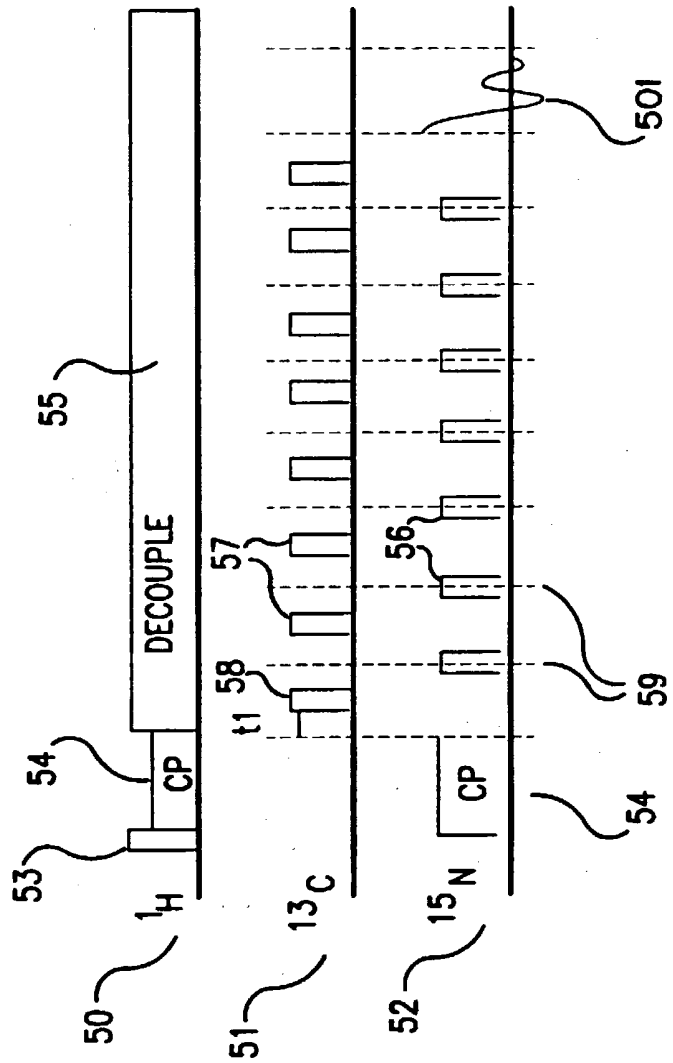

FIG.4

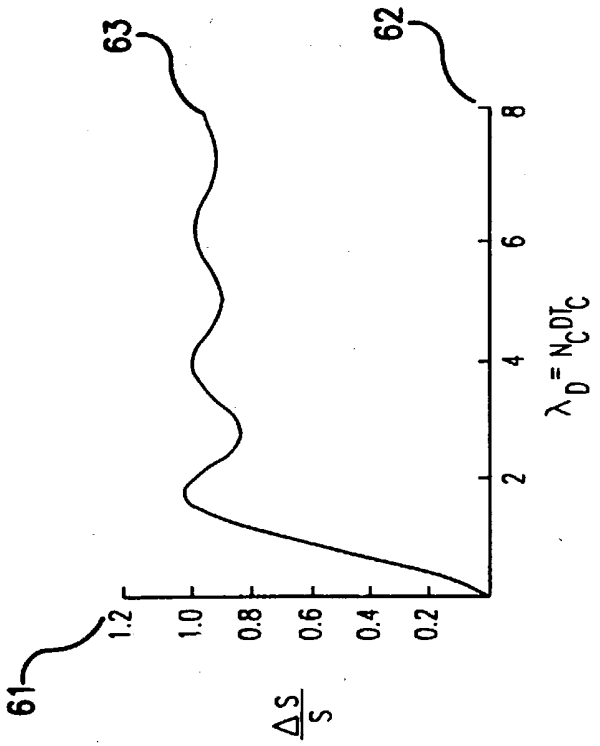

FIG.5

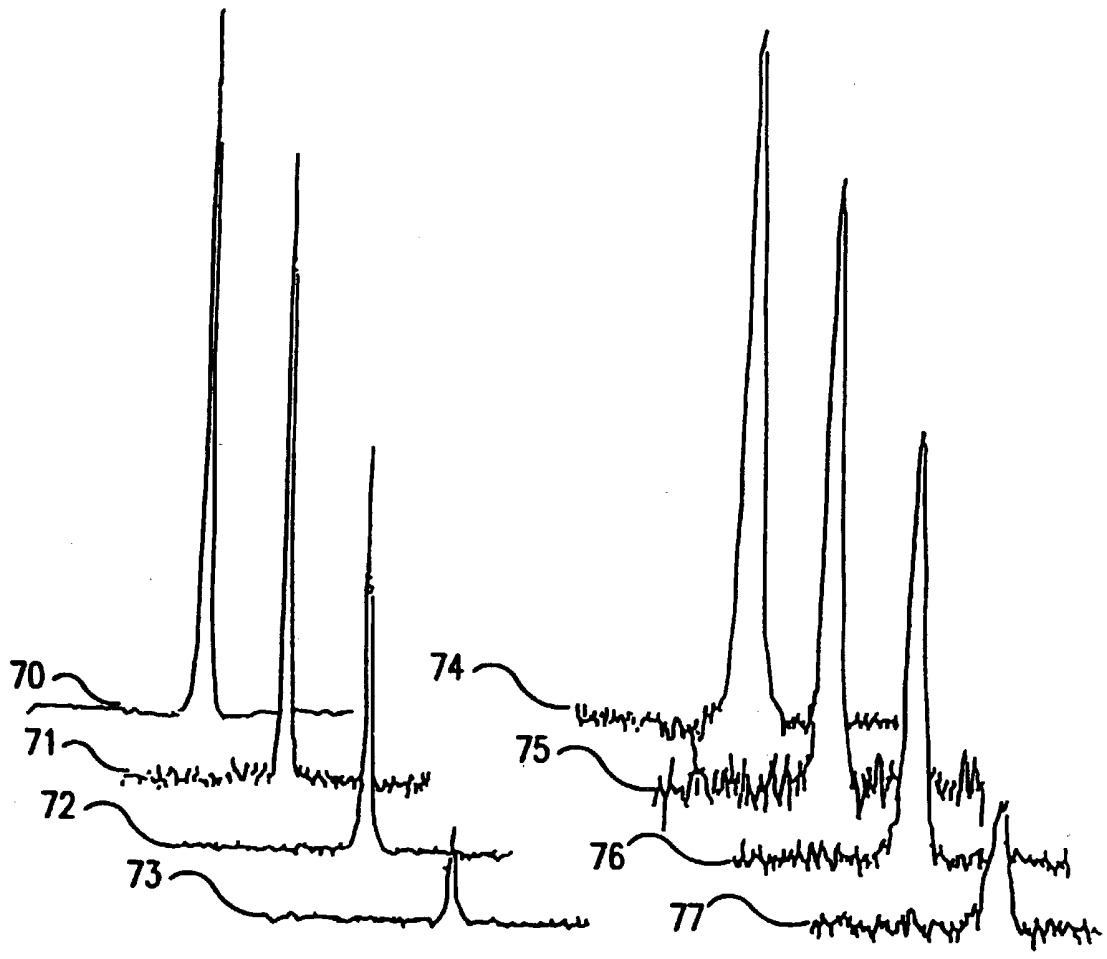

FIG.6

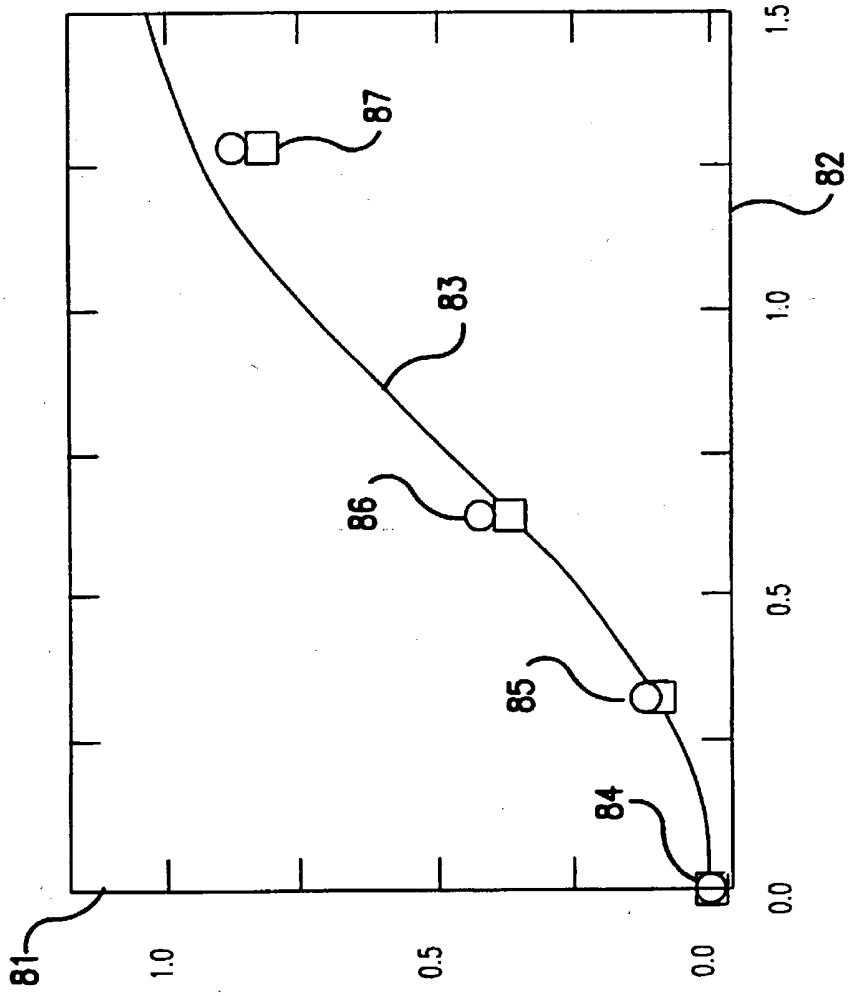

FIG.7

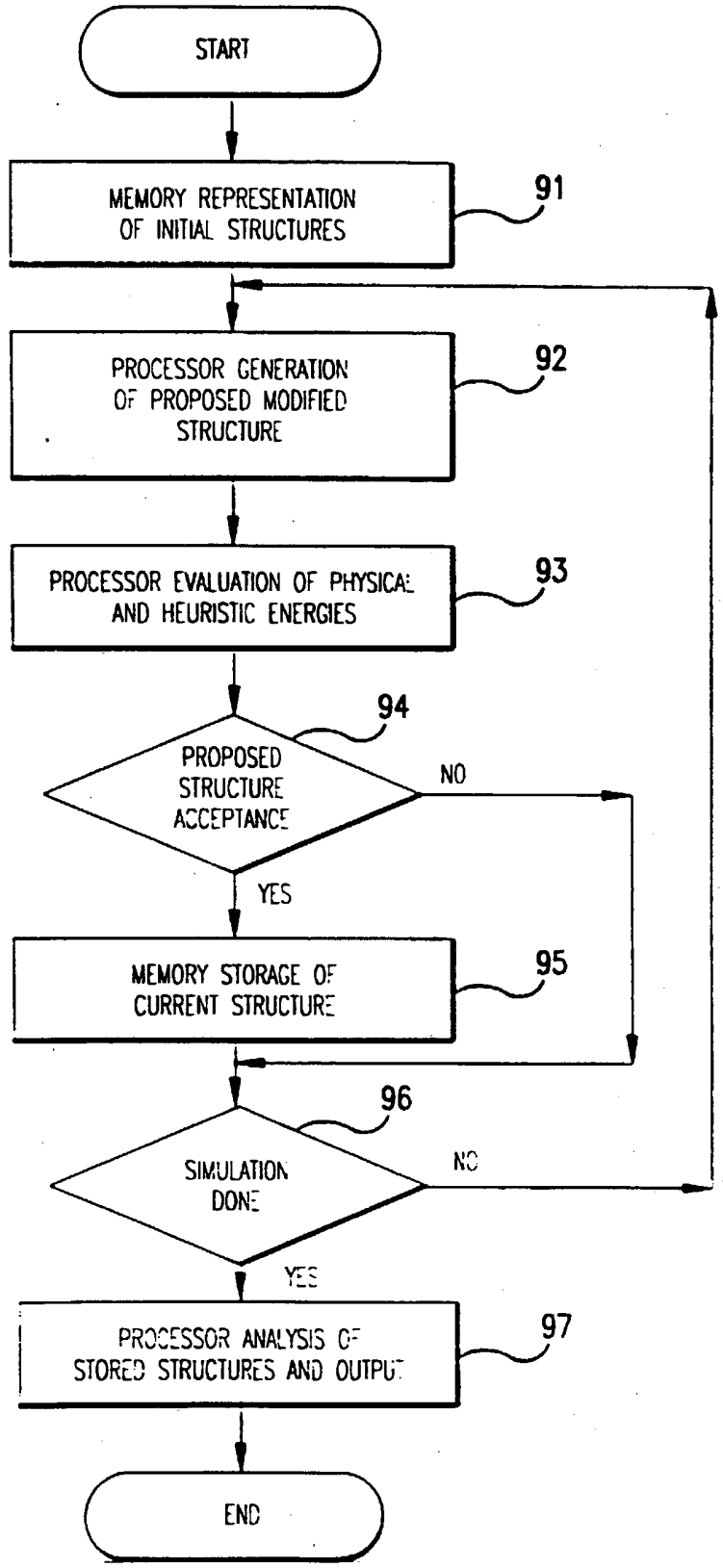
START
MEMORY REPRESENTATION OF INITIAL STRUCTURES
91
PROCESSOR GENERATION OF PROPOSED MODIFIED STRUCTURE
92
PROCESSOR EVALUATION OF PHYSICAL AND HEURISTIC ENERGIES
93
94
PROPOSED STRUCTURE ACCEPTANCE
NO
YES
MEMORY STORAGE OF CURRENT STRUCTURE
95
96
SIMULATION DONE
NO
YES
PROCESSOR ANALYSIS OF STORED STRUCTURES AND OUTPUT
97
END

FIG.8

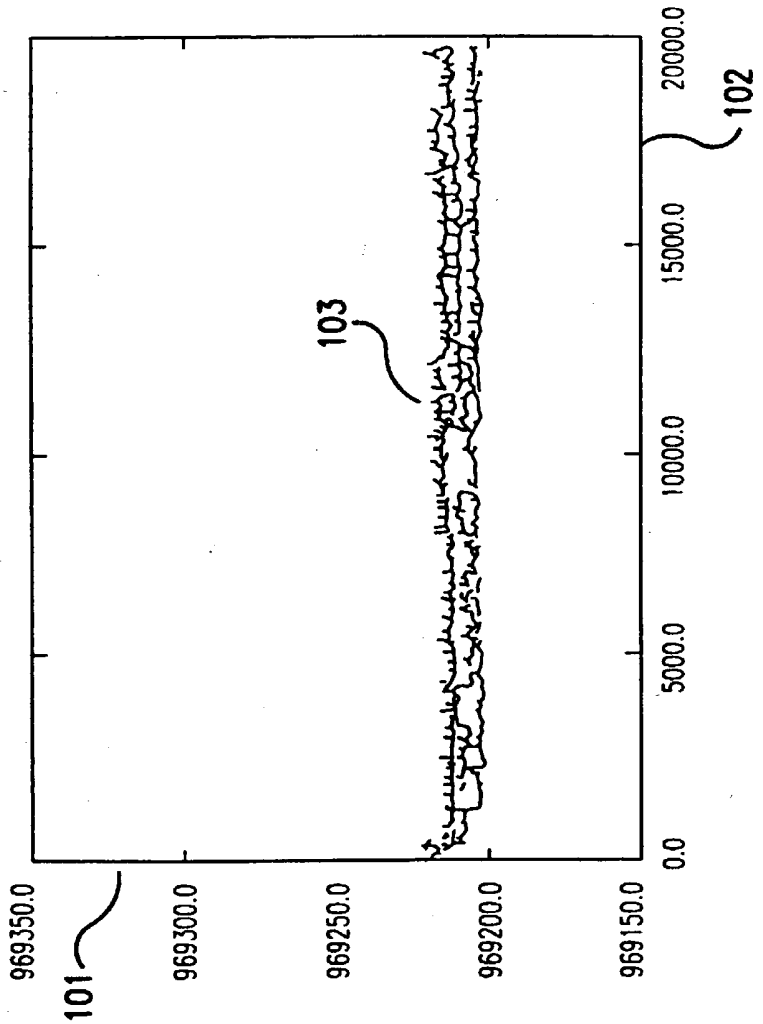
969350.0
969300.0
969250.0
969200.0
969150.0
101
0.0
5000.0
10000.0
15000.0
20000.0
102
103

FIG.9

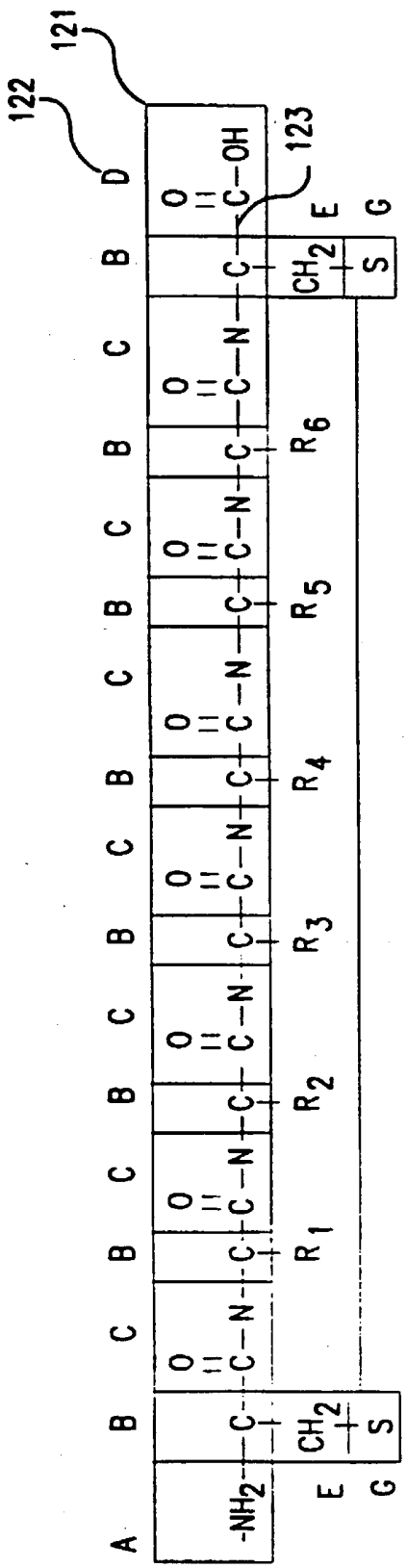

FIG.10

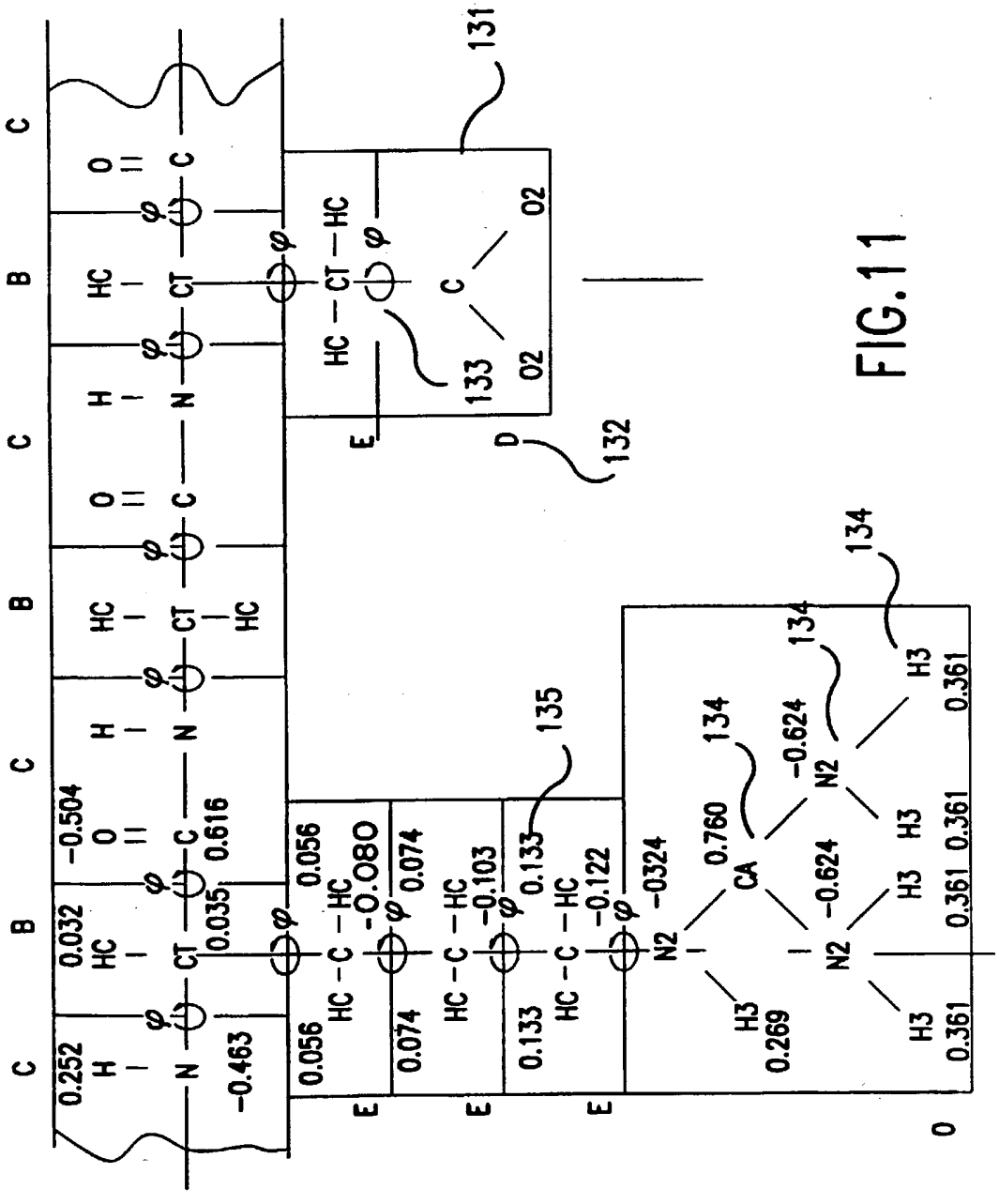
131
132
133
134
135
0.252
-0.463
0.032
-0.504
0.035
0.616
0.056
-0.080
0.074
-0.103
0.133
-0.122
-0.324
0.760
CA
-0.624
0.269
0.361

FIG. 11

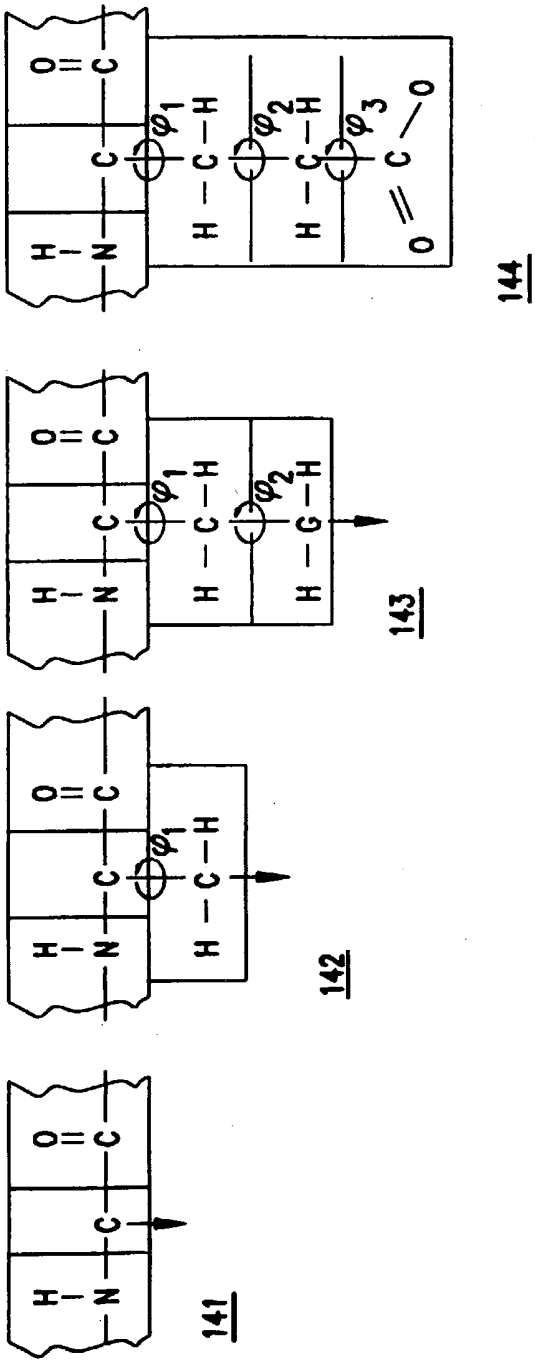

FIG. 12

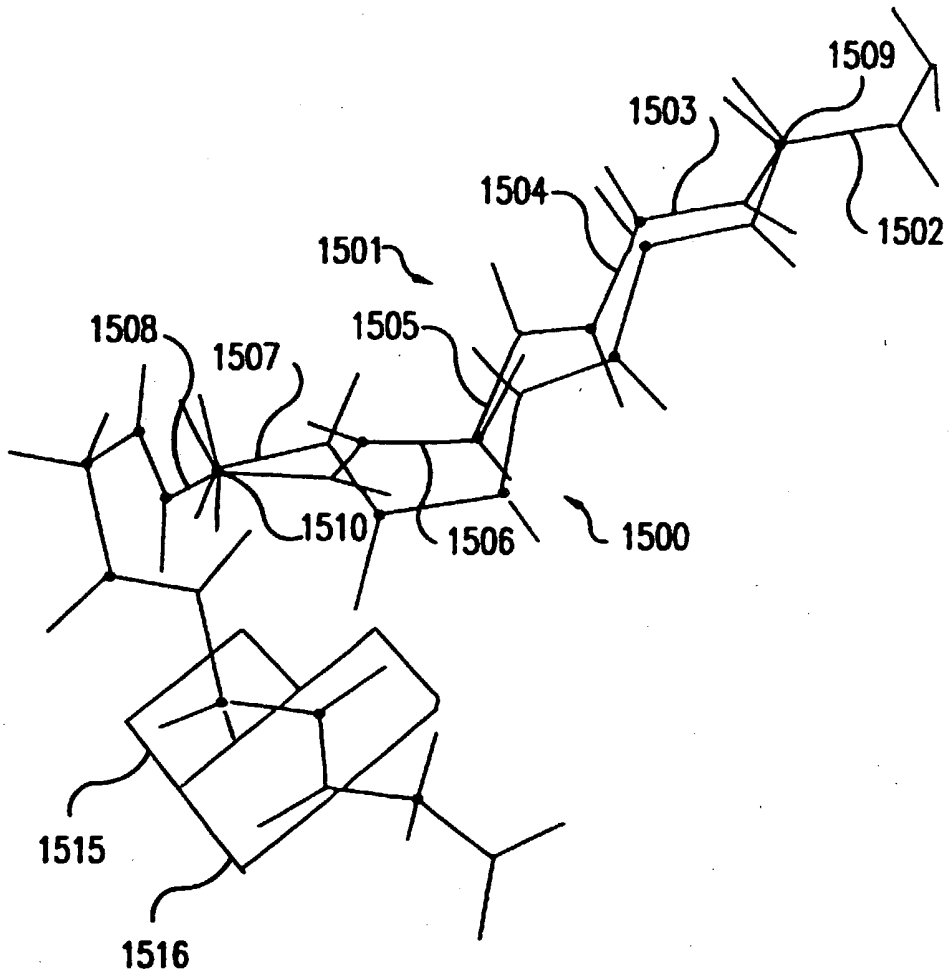
1509
1503
1504
1502
1501
1508
1505
1507
1510
1506
1500
1515
1516

FIG.13

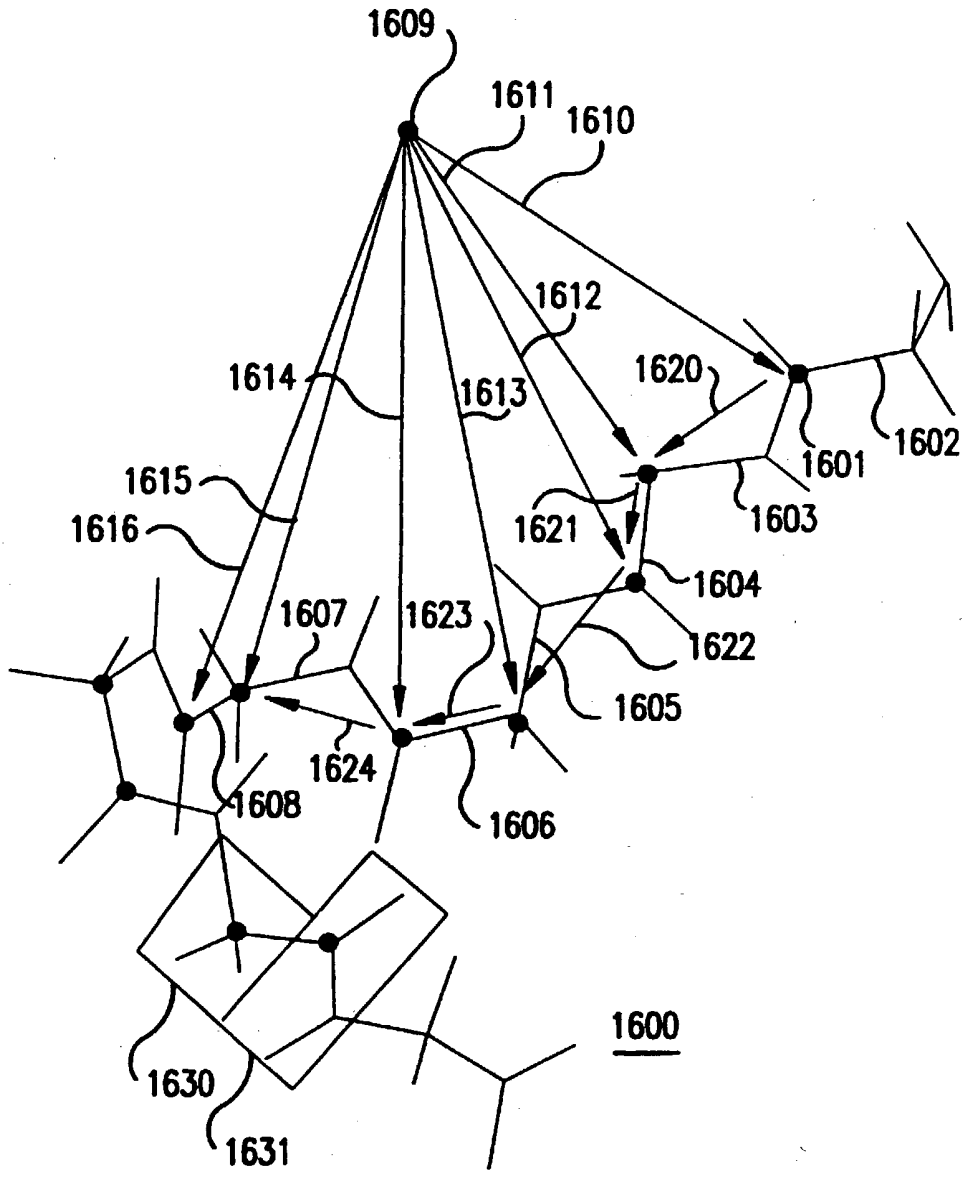
1609
1611
1610
1612
1614
1613
1620
1602
1601
1615
1603
1616
1621
1604
1607
1623
1622
1605
1624
1608
1606
1600
1630
1631

FIG. 14

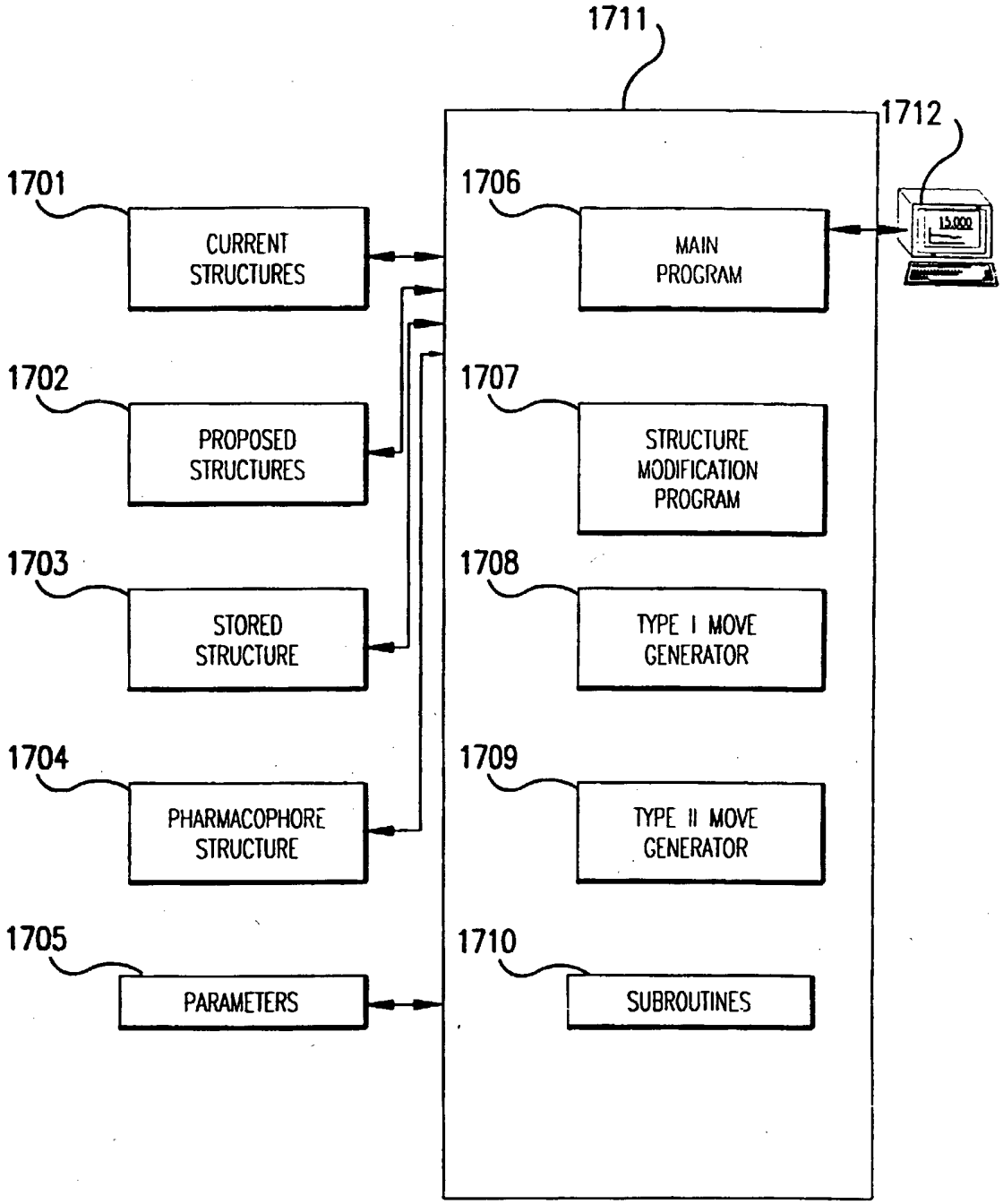
1711
1712
1701
CURRENT STRUCTURES
1706
MAIN PROGRAM
1702
PROPOSED STRUCTURES
1707
STRUCTURE MODIFICATION PROGRAM
1703
STORED STRUCTURE
1708
TYPE I MOVE GENERATOR
1704
PHARMACOPHORE STRUCTURE
1709
TYPE II MOVE GENERATOR
1705
PARAMETERS
1710
SUBROUTINES

FIG.15

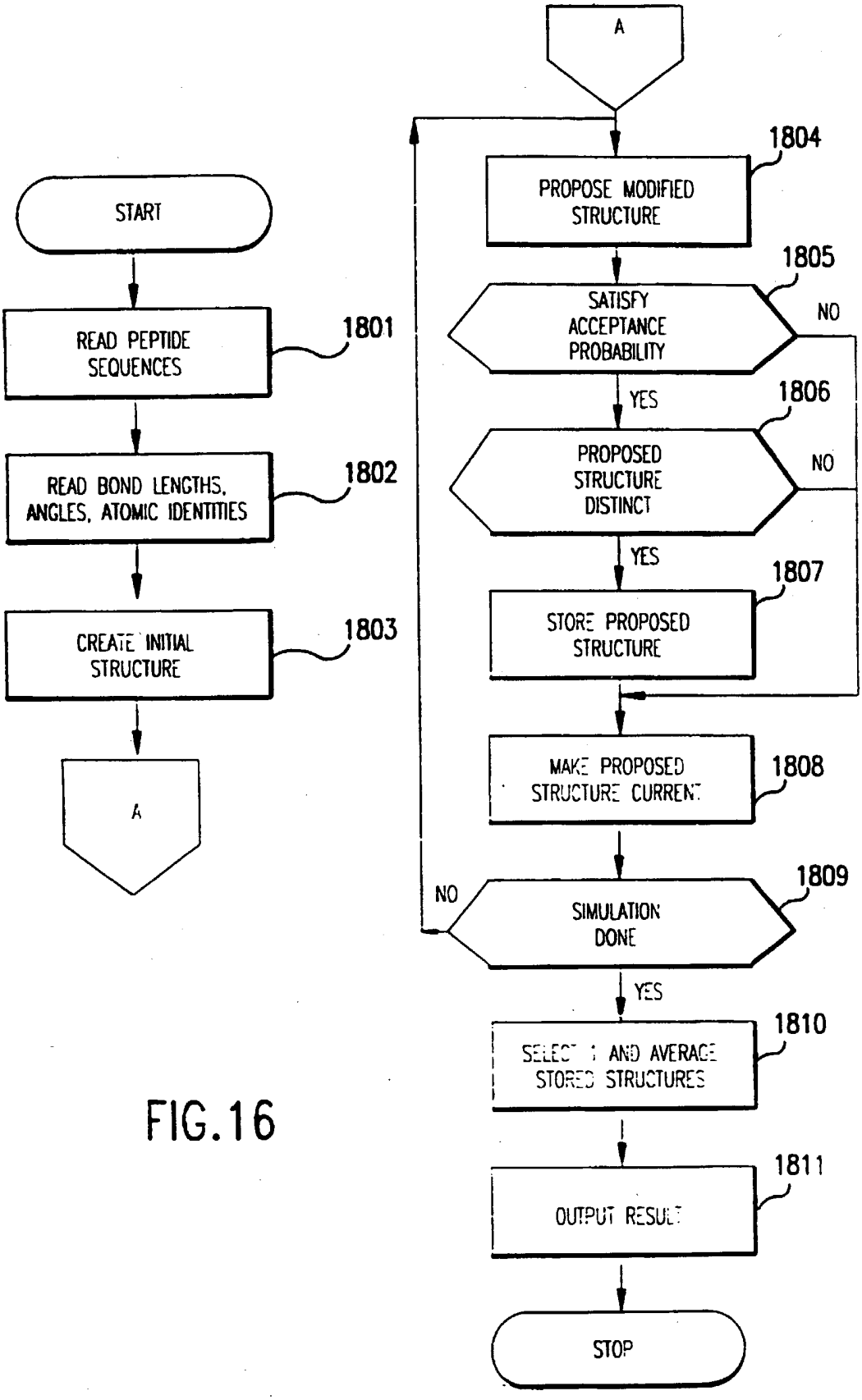
START
READ PEPTIDE SEQUENCES
1801
READ BOND LENGTHS, ANGLES, ATOMIC IDENTITIES
1802
CREATE INITIAL STRUCTURE
1803
A
A
PROPOSE MODIFIED STRUCTURE
1804
SATISFY ACCEPTANCE PROBABILITY
1805
NO
YES
PROPOSED STRUCTURE DISTINCT
1806
NO
YES
STORE PROPOSED STRUCTURE
1807
MAKE PROPOSED STRUCTURE CURRENT
1808
NO
SIMULATION DONE
1809
YES
SELECT 1 AND AVERAGE STORED STRUCTURES
1810
OUTPUT RESULT
1811
STOP

FIG.16

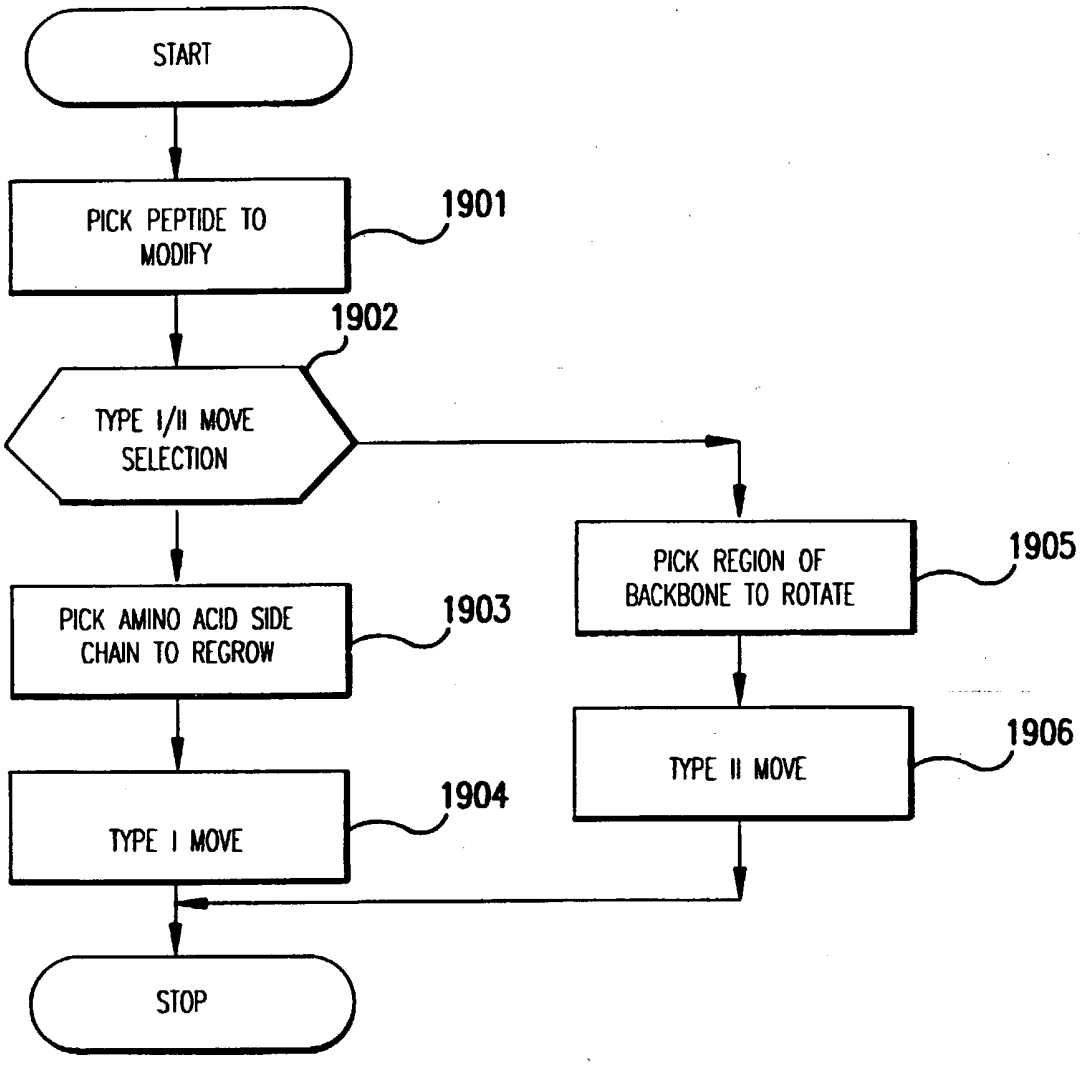
START
PICK PEPTIDE TO MODIFY
1901
1902
TYPE I/II MOVE SELECTION
PICK AMINO ACID SIDE CHAIN TO REGROW
1903
TYPE I MOVE
1904
PICK REGION OF BACKBONE TO ROTATE
1905
TYPE II MOVE
1906
STOP

FIG.17

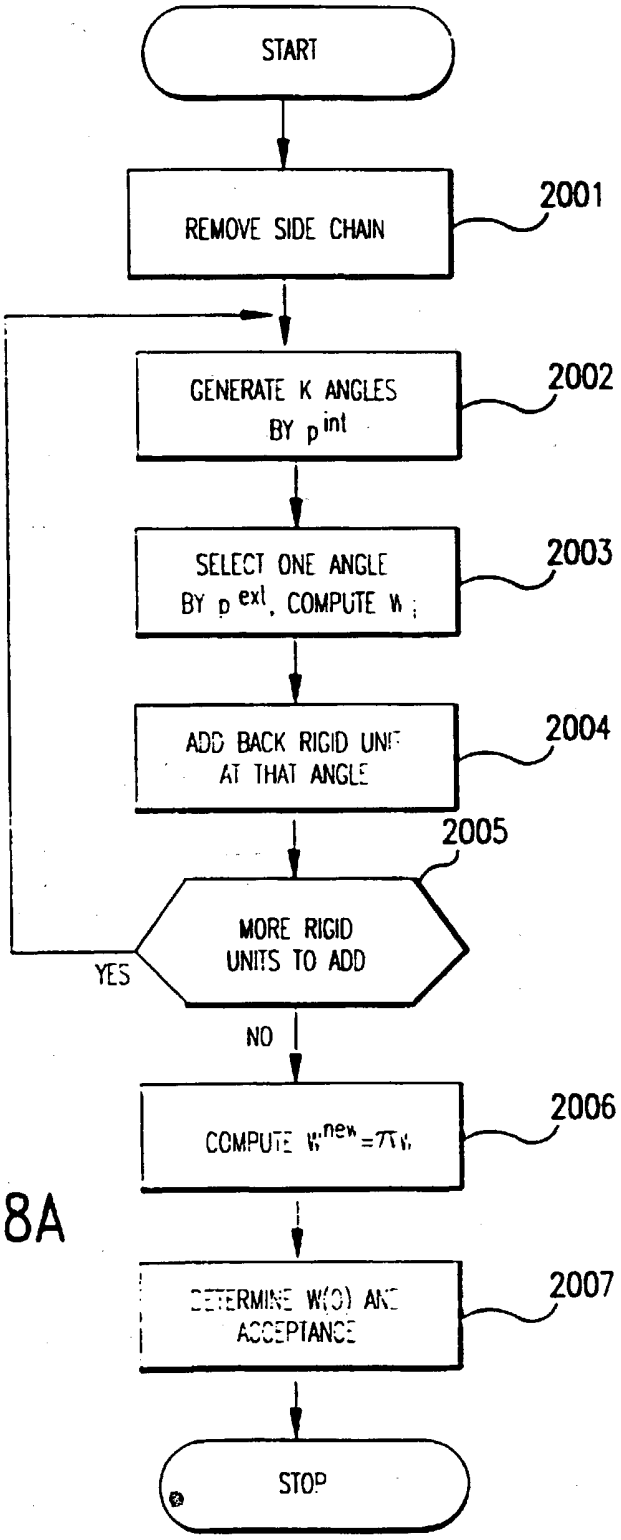
START
REMOVE SIDE CHAIN
2001
GENERATE K ANGLES BY p^{int}
2002
SELECT ONE ANGLE BY p^{exl}, COMPUTE W_i
2003
ADD BACK RIGID UNIT AT THAT ANGLE
2004
2005
MORE RIGID UNITS TO ADD
YES
NO
COMPUTE $W^{new}=\pi W_i$
2006
DETERMINE W(O) AND ACCEPTANCE
2007
STOP

FIG.18A

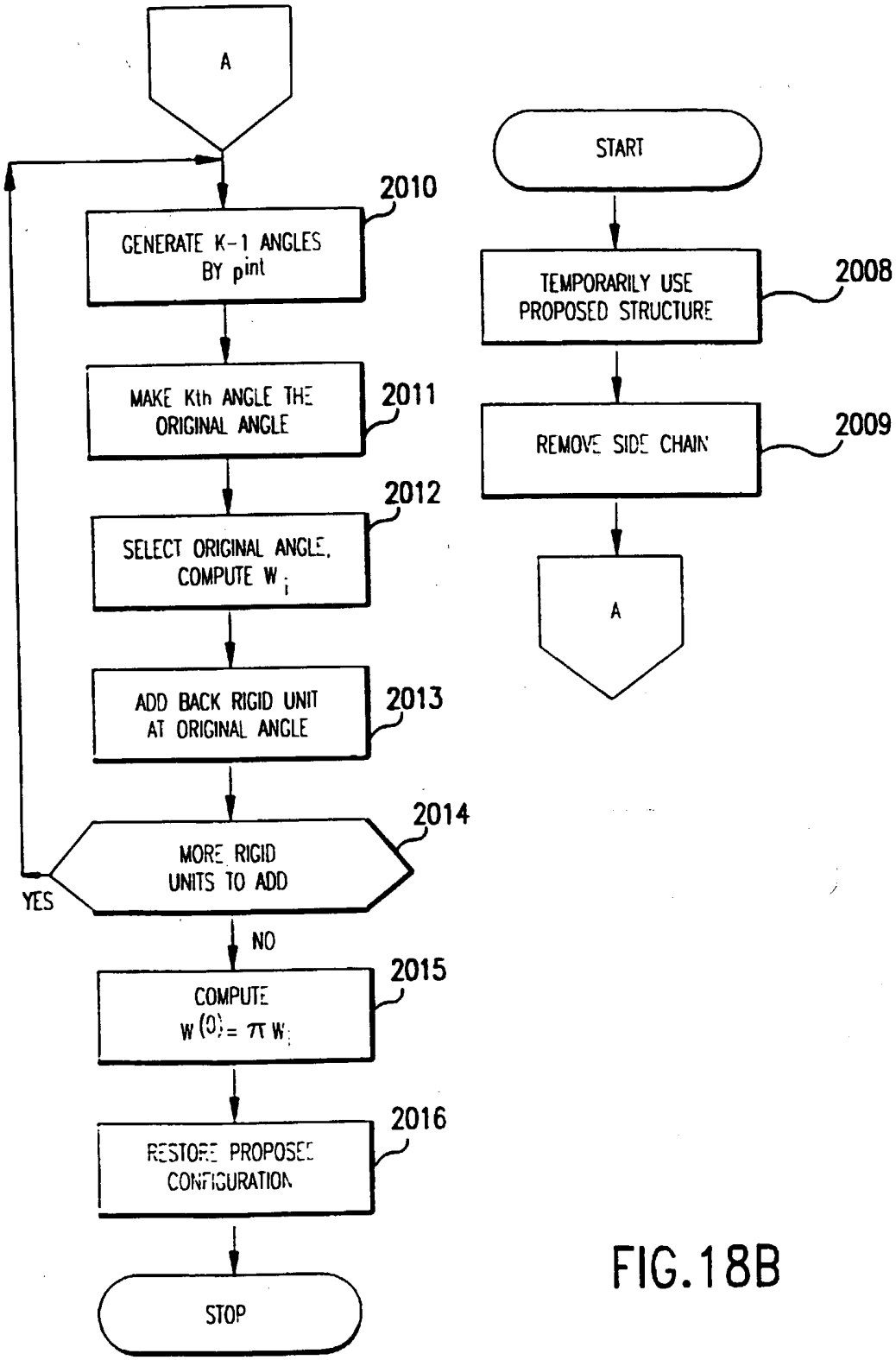
A
START
2010
GENERATE K-1 ANGLES BY pint
2008
TEMPORARILY USE PROPOSED STRUCTURE
2011
MAKE Kth ANGLE THE ORIGINAL ANGLE
2009
REMOVE SIDE CHAIN
2012
SELECT ORIGINAL ANGLE, COMPUTE W i
A
2013
ADD BACK RIGID UNIT AT ORIGINAL ANGLE
2014
MORE RIGID UNITS TO ADD
YES
NO
2015
COMPUTE W (0) = π W i
2016
RESTORE PROPOSED CONFIGURATION
STOP

FIG.18B

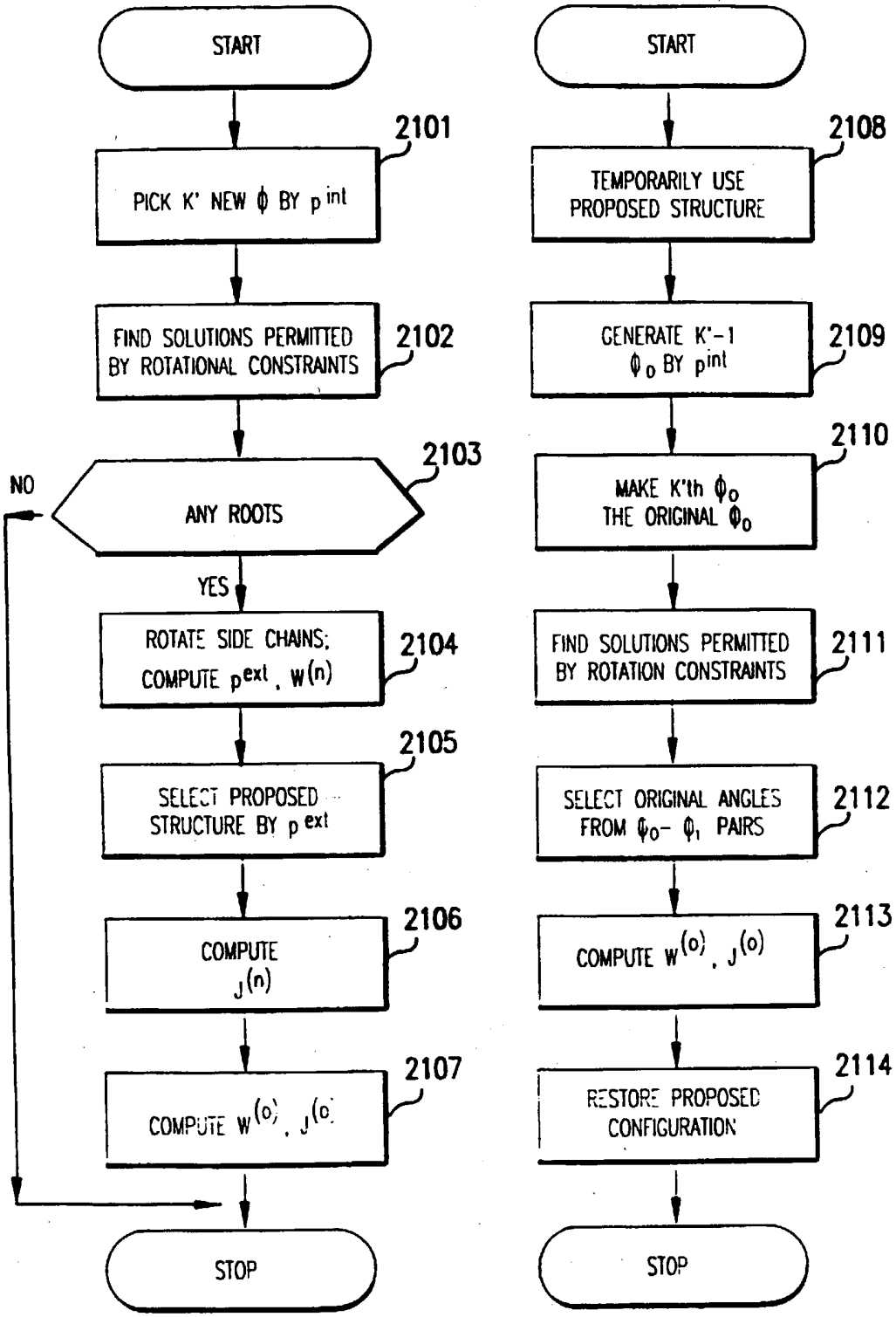

FIG.19A FIG.19B

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US96/04229

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) :G06F 17/50, 159:00; G06G 7/58; G01N 24/12, 33/53
US CL :364/413.01, 496, 578; 436/173, 501, 518; 435/7.1
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 364/413.01, 496, 578; 436/173, 501, 518; 435/7.1

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

APS, STN
search terms: library, pharmacophore, REDOR NMR, monte carlo

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X --- A	HODGKIN et al. A Monte Carlo Pharmacophore Generation Procedure: Application to the Human PAF Receptor. Journal of Computer Aided Molecular Design. 1993, Vol. 7, pages 515-534, see pages 517-521.	22-24, 26, 27, 51-106 --- 1-21, 25, 28-41, 108
X --- A	WILSON et al. The Calculation and Synthesis of a Template Molecule. Tetrahedron. 1993, Vol. 49, No. 17, pages 3655-3663, see entire document.	51-106 --- 1-41, 108
P, A	SEPETOV et al. Library of libraries: Approach to synthetic combinatorial library design and screening of "pharmacophore" motifs. Proceedings of the National Academy of Sciences. June 1995, Vol. 92, pages 5426-5430, see abstract.	1-41, 108

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
03 JULY 1996	25 JUL 1996
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer LORA M. GREEN Telephone No. (703) 308-0196

Form PCT/ISA/210 (second sheet)(July 1992)*

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US96/04229

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	BIANCHI et al. A Conformationally Homogenous Combinatorial Peptide Library. Journal of Molecular Biology. 1995, Vol. 247, pages 154-160, see abstract.	1-41, 108
X	US 5,265,030 (SKOLNICK ET AL.) 23 November 1993, see column 2, line 21-column 3, line 20.	68-102, 104-106
Y	GARBOW et al. Determination of the Molecular Confirmation of Melanostatin Using 13C,15N-REDOR NMR Spectroscopy. Journal of the American Chemical Society. 1993, Vol. 115, pages 238-244, see Experimental Section.	42-50, 107
Y	FERNANDEZ et al. Magnetic Resonance Studies of Polypeptides Adsorbed on Silica and Hydroxyapatite Surfaces. Journal of the American Chemical Society. 1992, Vol. 114, pages 9634-9642, see abstract.	42-50, 107

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US96/04229

Box I Observations where certain claims were found unsearchable (Continuation of item 1 of first sheet)

This international report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:

2. ☐ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:

3. ☐ Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box II Observations where unity of invention is lacking (Continuation of item 2 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

Please See Extra Sheet.

1. ☒ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.

2. ☐ As all searchable claims could be searched without effort justifying an additional fee, this Authority did not invite payment of any additional fee.

3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest

☐ The additional search fees were accompanied by the applicant's protest.

☐ No protest accompanied the payment of additional search fees.

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US96/04229

BOX II. OBSERVATIONS WHERE UNITY OF INVENTION WAS LACKING

This ISA found multiple inventions as follows:

This application contains the following inventions or groups of inventions which are not so linked as to form a single inventive concept under PCT Rule 13.1. In order for all inventions to be examined, the appropriate additional examination fees must be paid.

Group I, claim(s) 1-41 and 108, drawn to a method of determining a consensus pharmacophore structure.
Group II, claim(s) 42-50 and 107, drawn to a method of making solid state magnetic resonance methods.
Group III, claim(s) 51-67 and 103, drawn to a method of configurational Monte Carlo determination.
Group IV, claims 68-102 and 104-106, drawn to an apparatus for configurational bias Monte Carlo determination.

The inventions listed as Groups I-IV do not relate to a single inventive concept under PCT Rule 13.1 because, under PCT Rule 13.2, they lack the same or corresponding special technical features for the following reasons:

Groups I and Groups II and III do not share a special technical feature as each has different steps and different end results. The method of Group I requires screening of a diversity library and generation of a pharmacophore, neither of which is required by the methods of Groups II and III. The distances required for the pharmacophore generation of Group I could be obtained by methods other than through the use of solid-state NMR methods ofGroup II, such as by solution NMR or X-ray crystallography. In addition, the method of Group I as claimed does not require the method of Group III as claimed, as the dependant claim of Group I which specifies the Monte Carlo method (claim 24) requires generating a proposed structure based on data for diversity libraries, whereas the method of Group II does not require the use of diversity libraries. Thus, Groups II and III lack the special technical feature of Group I, i.e. using a diversity library to generate a consensus pharmacophore structure.

Groups II and III are related as separate methods, as Group II is drawn to a method of making NMR measurements, while the method of Group III is drawn to a method of configurational monte carlo analysis. Thus, Groups II and III do not share a technical feature.

Groups I and II are related to Group IV as separate methods and product, as the methods of Groups I and II as claimed do not require the apparatus of Group IV as claimed.

Groups III and IV are related as separate method and product. The method of Group III as claimed does not require the use of the apparatus of Group IV as claimed. In addition, the apparatus of Group IV could be used in methods other than the method of Group III such as use generating structures using NMR data obtained from a compound in solution phase.